Departamento de Informática

PROYECTO FIN DE CARRERA

Aplicación Web para la gestión de redes inalámbricas de sensores ubicados en Google Maps.

Autora: Gloria Rivero Ortega

Tutor: Alejandro Calderón Mateos

Leganés, octubre de 2010

Título: Aplicación web para la gestión de redes inalámbricas de sensores ubicados en Google Maps.

Autor: Gloria Rivero Ortega

Director: Alejandro calderón Mateos

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Si se quiere ascender por cuestas empinadas,

es necesario al principio andar despacio

(William Shakespeare)

Resumen

Este proyecto describe los métodos y tecnologías a seguir para construir una aplicación web que gestione redes inalámbricas de sensores posicionados en mapas, utilizando las herramientas de Google Maps. Los datos obtenidos por los sensores serán almacenados en una base de datos y representados en la aplicación mediante gráficas.

Abstract

This project describes the methods and technologies for building a Web application in order to manage Wireless Networks of Sensors positioned by using Google Maps tools. Data collected by the sensors will be stored in a database, and then they will be represented in the Web application with graphical views.

Índice de contenidos

Índice de Ilustraciones

Índice de tablas

Capítulo 1

Introducción y objetivos

Esta sección tiene el objetivo de presentar una visión global sobre el proyecto realizado, así como indicar las motivaciones y metas por las que se ha decidido realizar este proyecto. Para finalizar con este punto se mostrará una lista de los acrónimos utilizados en el documento con sus correspondientes definiciones para ayudar a la comprensión del contenido.

1.1 Visión general

El objetivo del proyecto consiste en el desarrollo del prototipo de una aplicación web integrada con una base de datos, a través de la cual, se representarán de manera gráfica los datos almacenados.

La idea principal reside en almacenar en una base de datos aquellos valores recogidos por una red de sensores, donde cada nodo se encuentra estratégicamente situado en diferentes puntos geográficos y representar, en la aplicación, tanto la posición de los sensores como los datos recogidos por los mismos de una manera gráfica que ayuda a la rápida comprensión por parte del usuario.

La aplicación realiza una distinción entre zonas rurales o urbanas, dependiendo de la localización de los sensores, puesto que cada una de las mismas necesitará de unos cuidados diferentes dependiendo de los resultados recogidos. Cada una de las zonas se encuentra reflejada en la aplicación gracias a las herramientas de Google con las que se ha trabajado en todo el proyecto.

En el proyecto se han utilizado las herramientas de Google Maps para posicionar los sensores en los mapas y así conseguir un resultado más visual, además de utilizar herramientas de *JPGraph* para las representaciones gráficas.

1.2 Motivación

Con este proyecto se pretende dar una visión de posibles nuevas herramientas para controlar, almacenar y visualizar los datos recogidos por una red de sensores de manera que facilite la gestión a los usuarios.

Las mejoras en las comunicaciones inalámbricas y los grandes avances en la fabricación de sensores para mediciones de todo tipo han promovido el desarrollo de este tipo de redes experimentando un gran avance que se ha visto traducido a su implantación en numerosos ámbitos. Además, son sistemas que ofrecen una gran escalabilidad a bajo coste, por lo que se convierten en la opción perfecta a utilizar cuando se pretenden monitorizar distintos variables de entorno cubriendo un amplio espacio de terreno.

Este tipo de redes tienen un ciclo de vida muy largo y generan en un corto espacio de tiempo una gran cantidad de nuevos valores, lo que hace necesario la inclusión en el sistema de una base de datos escalable para conseguir almacenar toda la información generada por la red de una manera fiable permitiendo además, su posterior tratamiento.

Debido a esto, en la actualidad existen numerosas líneas de investigación que versan sobre este tipo de redes así como hablan sobre las posibles soluciones a la hora de almacenar y representar el gran volumen de datos generado de manera que resulte de utilidad para el usuario y que facilite la interpretación de estos datos a través de una interfaz atractiva. De aquí nace la motivación que me ha conducido a que el presente proyecto trate de generar un nuevo valor añadido a través de la geolocalización de los sensores y su presentación en un mapa interactivo.

1.3 Objetivos

El objetivo principal de este proyecto es investigar sobre posibles soluciones de representación para el problema que supone tratar el gran volumen de datos proporcionado por una red de sensores inalámbricos durante ciclos de vida muy largos.

Estudiando las posibles herramientas disponibles para llevar a cabo el almacenamiento y representación de los datos, se han propuesto una serie de objetivos más concretos para la realización del prototipo de una aplicación que hará posible el objetivo principal del proyecto:

- Diseño y desarrollo de una aplicación web dinámica que muestre de forma clara la información en tiempo real de una red inalámbrica de sensores

- Estudio de las posibilidades que ofrece Google para insertar sus herramientas en aplicaciones web privadas.

- Búsqueda de soluciones a las limitaciones de desarrollo de las herramientas de Google.

- Implementación de una base de datos estable que sea capaz de actualizarse de forma automática con los datos proporcionados por una red de sensores.

- Realización de gráficos usando los datos almacenados en una base de datos.

1.4 Acrónimos

AJAX: *Asynchronous Javascript And XML*

API: *Application Programming Interface*

CSS: *Cascading Style Sheets*

FastCGI: *Fast Common Gateway Interface*

FTP: *File Transfer Protocol*

GPL: *General Public License*

HTML: *HyperText Markup Language*

HTTP: *HyperText Transfer Protocol*

PHP: *Hypertext Pre-Processor*

RFID: *Radio Frecuency Identification*

SCGI: *Simple Common Gateway Interface*

SGBD: *Sistema Gestor de Bases de Datos*

SRC: *Source*

SSL: *Secure Sockets Layer*

SQL: *Structured Query Language*

URL: *Uniform Resource Protocol*

W3C : *World Wide Web Consortium*

XML: *eXtensible Markup Language*

1.5 Estructura del documento

Este apartado está destinado explicar brevemente los puntos a desarrollar en los capítulos de este documento.

Capitulo 1:

Se pretende dar una visión global del proyecto realizado, los objetivos establecidos, y los acrónimos usados en el resto del documento para una mayor comprensión.

Capitulo 2:

En el Estado de la Cuestión se tratarán temas relacionados con las redes de sensores inalámbricas y las tecnologías estudiadas para hacer posible el desarrollo del proyecto.

Capitulo 3:

Este capítulo muestra las herramientas y técnicas desarrolladas en las fases de análisis y de diseño del proyecto. Además se mostrarán los aspectos más destacados llevados a cabo en la fase de implementación.

Capitulo 4:

En Planificación y Presupuesto de analizará el Diagrama de Gantt, método utilizado para la planificación, y se desglosarán los costes que han hecho posible el desarrollo del proyecto.

Capitulo 5:

Conclusiones y Trabajos Futuros, es un capítulo subjetivo acerca de los objetivos conseguidos con la elaboración de este proyecto así como las posibles mejoras a realizar en trabajos futuros. Además , este apartado incluye la Bibliografía utilizada.

Capitulo 6:

Este capítulo contiene los anexos realizados para, en caso de ser necesario, llegar a un mejor entendimiento del funcionamiento del servidor utilizado, de la base de datos, y de la aplicación web.

Capítulo 2

Estado de la Cuestión

En este apartado se estudian y analizan los sistemas gestores de bases de datos y los servidores web así como los estándares más utilizados en la actualidad dependiendo de la naturaleza de su contenido (estático y dinámico) y de las estructuras que presentan, con esto se pretende explicar los motivos de las decisiones tomadas acerca de las tecnologías aplicadas en el desarrollo del proyecto. Además se hablará brevemente de los lenguajes que se han utilizado en la implementación.

2.1 Estudio Previo

2.1.1 Estudio de mercado

Las investigaciones en el campo de las redes de sensores inalámbricas predicen la llegada de nuevas generaciones de sensores inteligentes capaces de organizarse e interconectarse de forma independiente.

En los últimos años se está apostando fuerte por esta tecnología y se habla de una importante revolución tecnológica de los sensores. Hoy en día existen en el mercado distintas alternativas para gestionar las redes de sensores:

- **Monitoreo Costero de Temperatura**[1]: Trabajo financiado por la Secretaría de Ciencia y Técnica de la Universidad Nacional de la Patagonia San Juan Bosco, donde se realizaron pruebas experimentales para emplear sistemas analógicos digitales reconfigurables.

- **NI wireless sensor network**[2]: desarrollado por la empresa *National Instruments* cuyo fin principal es la venta de sus sistemas de sensores para implantar una posible red.

- **Code Blue**: desarrollado por la universidad de Harvard implementar redes de sensores en el campo de la medicina. Consiste en una plataforma hardware que permite enviar y recibir datos médicos de un paciente de forma inalambrica.

Como conclusión al estudio de mercado, se puede afirmar que se están realizando en la actualidad, numerosas investigaciones para mejorar la tecnología de los sensores y de la comunicación entre ellos, pero todavía no existen aplicaciones para representar de forma gráfica los datos y ayudar al usuario en la interpretación de los mismos.

[1] *http://ingenieria.unlam.edu.ar/uea2010/trabajos/uea2010_submission_58.pdf*
[2] http://www.ni.com/wsn/

2.1.2 Redes de sensores inalámbricas

Las redes de sensores consisten en la composición de pequeños dispositivos provistos de sensores, cada uno de los cuales se denomina nodo, y todos ellos se agrupan para realizar una tarea común. Cada uno de los dispositivos miden factores de su entorno, bien sean de humedad, temperatura, luminosidad...etc., y envían a un procesador central los datos recogidos para ser tratados. El conjunto de sensores que realizan un mismo fin y se comunican entre ellos mediante protocolos de comunicación, sin cables, se denomina redes de sensores inalámbricas.

Este tipo de redes están diseñadas para enviar los datos medidos a una unidad central de procesamiento mediante el uso de distintas tecnologías sin cable. Las tecnologías más utilizadas en la comunicación de los sensores inalámbricos son el Bluetooth, la identificación de la frecuencia de radio (RFID) o el estándar IEE 802.11.

Los nodos de la red deben ser capaces de ejercer tanto de emisores como de receptores, del mismo modo, el nodo que actúa de puente con el servidor que procesa los datos también actúa de emisor y receptor.

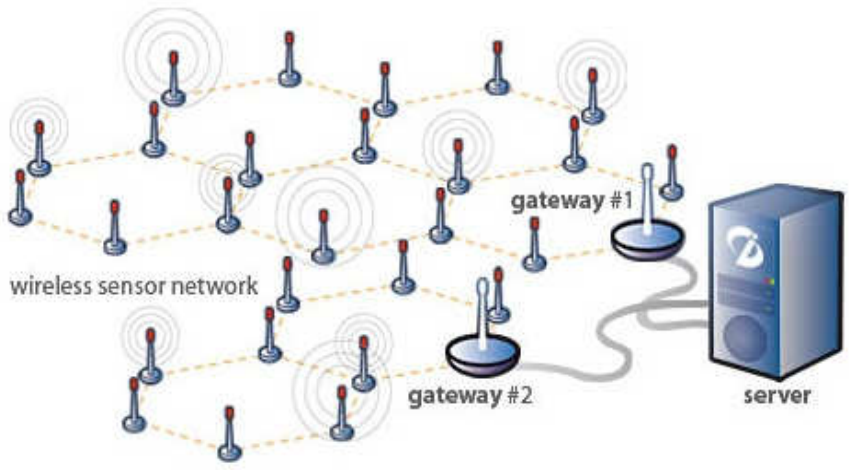


Ilustración 1: Ejemplo de red de sensores inalámbrica

Este tipo de redes de sensores se caracterizan por su sencillez a la hora de desarrollarlas e implantarlas, puesto que como hemos dicho anteriormente, cada uno de los nodos puede ser configurado para realizar tareas de emisor, receptor o de enrutador. Las redes de sensores inalámbrica consta de numerosas virtudes, algunas a destacar son:

- **Uso eficiente de energía**: Son redes que necesitan de un consumo mínimo de energía para su funcionamiento, lo que permite una mayor autonomía.

- **Alta disponibilidad**: Este tipo de redes están configuradas mediante algoritmos de enrutamiento dinámico, lo que permite que la red siga funcionando en caso de que alguno de los nodos fallara o fuera extraído de la red.

- **Escalabilidad**: Las redes pueden ser ampliadas en cualquier momento de una manera fácil, pues si se añaden nuevos nodos a la red, el resto los reconocen automáticamente y sólo se tendrían que actualizar las tablas de rutas.

- **Fácil implantación**: Al realizar las tareas de comunicación de forma inalámbrica se facilita la implantación de este tipo de redes, posibilitando su uso en un gran número de escenarios.

- **Bajo coste**: Este tipo de nodos se fabrican masivamente, puesto que se requiere de un gran número para la red de sensores, y sus costes no son elevados, además al tener un uso eficiente de energía el consumo y los costes de mantenimiento son mínimos.

Gracias a todas estas características se permite implantar redes de sensores inalámbricas en muy diferentes entornos que ayudarán al buen mantenimiento de los espacios, como las zonas agrícolas, sociales, civiles, sanitarias...etc.

2.1.3 Sistemas gestores de bases de datos

Los sistemas de gestión de bases de datos son un tipo de software específico dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones web que la utilizan. Estos sistemas se encargan de manejar los datos que se van a utilizar en las aplicaciones de una manera clara, sencilla y ordenada.

Los objetivos de utilizar un SGBD son abstraer la información, conseguir mayor seguridad, facilidad en el manejo de transacciones y obtener una independencia y consistencia de los datos. De este modo se ahorra al usuario detalles acerca del almacenamiento físico de los datos y se garantiza que la información se encuentra segura para todos los usuarios y se respetan los permisos asignados a cada uno.

Además de ventajas para los usuarios, los SGBD facilitan la manipulación de grandes volúmenes de datos y la programación para los desarrolladores, por tanto, bajan drásticamente los tiempos de desarrollo y aumentan la calidad del sistema. Como inconvenientes solo destacaremos la necesidad de disponer de administradores de la base de datos, lo que puede incrementar los costes de operación así como un coste adicional para el hardware necesario, ya que los requisitos de hardware para correr un SGBD, por lo general, son relativamente altos.

De los posibles SGBD libres disponibles en el mercado se han analizado las características de los más utilizados, MySQL, SQL Server y Oracle. A continuación se muestra una tabla esquemática con ventajas y desventajas que han ayudado a la elección de MySQL para el proyecto.

	MySQL[3]	SQL Server[4]	Oracle[5]
S.O.	Multiplataforma	Microsoft Windows	Multiplataforma
Licencia	GPL	Microsoft EULA	Privada
Relacional	SI	SI	SI
Multihilo	SI	SI	SI
Multiusuario	SI	SI	SI
Disparadores	SI	SI	SI
Claves ajenas	SI	SI	SI
Vistas actualizables	SI	SI	SI
Procedimientos almacenados	SI	SI	SI
Escalabilidad	NO	SI	SI
Estabilidad	NO	SI	SI
Conectividad segura	SI	SI	SI

Tabla 1: Comparativa de sistemas gestores de bases de datos

[3] *www.mysql.com/*

[4] *www.microsoft.com/sqlserver/2008/en/us/*

[5] *www.oracle.com/es/index.html*

Como conclusiones podemos decir que MySQL es ligero y relativamente rápido, pero algo escaso de características, mientras que SQL Server es más robusto y aunque tiene un buen rendimiento, es más lento que MySQL. Por otro lado para utilizar SQL Server hay que pagar las licencias de Microsoft, en cambio, MySQL es gratuito, ya que se ofrece bajo la GNU GPL (*GNU General Public License*) para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privados, deben comprar una licencia específica que les permita este uso.

SQL Server es capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Sus principales ventajas son la escalabilidad, estabilidad y seguridad, además, permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red solo acceden a la información.

La tecnología Oracle se encuentra prácticamente en todas las industrias alrededor del mundo, es el proveedor mundial líder de software para administración de información, y la segunda empresa de software. Oracle está considerado uno de los sistemas de bases de datos más completos. A pesar de todo esto el coste de las licencias para su uso es elevado, este ha sido el principal motivo para no utilizar este SGBD.

En la aplicación web a desarrollar hay baja concurrencia en la modificación de los datos pero el entorno es intensivo en lecturas, lo que hace que MySQL sea ideal para cumplir con nuestros requisitos, además de que al tener las licencias gratuitas el acceso a la plataforma es más fácil.

2.1.4 Servidores Web

Un servidor web es un programa diseñado para transferir hipertextos, páginas web o páginas HTML, implementando el protocolo HTTP. El servidor web se ejecuta en un ordenador manteniéndose a la espera de peticiones por parte de un cliente (navegador web) y responde a estas peticiones adecuadamente, mediante una página web que se mostrará en el navegador o mostrando el respectivo mensaje si se detectó algún error.

Además de la transferencia de código HTML, los servidores web pueden entregar aplicaciones web. Éstas son porciones de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Hay que distinguir entre:

- ***Aplicaciones en el lado del cliente***: El cliente web es el encargado de ejecutarlas en la máquina del usuario. El servidor proporciona el código de las aplicaciones al cliente y éste las ejecuta a través del navegador. Para esto es necesario que el cliente disponga de un navegador con capacidad para ejecutar aplicaciones.

- ***Aplicaciones en el lado del servidor***: El servidor web ejecuta la aplicación y una vez ejecutada se genera un código HTML, el servidor toma el código recién creado y lo envía al cliente por medio del protocolo HTTP.

El término servidor también se utiliza para referirse al ordenador físico en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizarlos. Este uso dual puede llevar a confusión, un servidor web podría referirse a la máquina que almacena y maneja los sitios web y también podría aludir al software que funciona en la máquina y controla la entrega de los componentes de las páginas web como respuesta a las peticiones de los navegadores.

Existen empresas dedicadas al "*hosting*", es un negocio que consiste en alojar, servir y mantener archivos para uno o más sitios web. La mayoría de los servicios de *hosting* ofrecen conexiones que para una persona individual resultarían muy costosas. Usar este servicio permite que muchas compañías compartan el coste de una conexión rápida a internet para el acceso a los archivos de sus sitios web.

Hay una amplia variedad de servicios de *hosting*. El más básico es el de archivos donde se pueden alojar las páginas de los sitios web y otros archivos vía ftp o una interfaz web. Normalmente no se requiere de grandes inversiones de dinero para alojar las páginas web en servidores, existen algunos gratuitos que aunque tienen límites muy grandes de espacio y de tráfico, comparado con el de pago, son suficientes para pequeñas aplicaciones.

Otra opción es instalar un servidor web en nuestro ordenador para poder montar una página web propia sin necesidad de contratar *hosting*. El problema de usar nuestro ordenador como servidor web local es que se debería tener encendido permanentemente para que esté accesible de forma continua, como el resto de sitios web, y el coste de recursos y electricidad sería importante.

Algunos de los servidores web más importantes son Apache[6] y Cherokee[7]. Apache es el servidor con más años de experiencia en el mundo de los desarrolladores web, produciendo todo tipo de soluciones de software para aplicaciones muy diversas. En los últimos años se está oyendo hablar de Cherokee como una alternativa viable a Apache, más rápida y fácil de usar.

Apache es un servidor web destacado por jugar un papel importante en el crecimiento inicial de la *World Wide Web*. Es desarrollado y mantenido por una comunidad abierta de desarrolladores. La aplicación está disponible para una amplia variedad de sistemas operativos, se caracteriza por ser código abierto.

[6] *www.**apache**.org*

[7] *www.**cherokee**-project.com*

Cherokee es muy rápido, flexible y fácil de configurar. Es compatible con las tecnologías más usadas hoy en día: FastCGI, SCGI, uWSGI, SSI TLS y conexiones cifradas SSL, hosts virtuales, la autenticación, la codificación de ventanas, de equilibrio de carga, streaming de video...etc.

Apache se utiliza principalmente para servir contenido estático y dinámico de páginas web en la *World Wide Web*. Es un servidor web usado para muchas tareas, donde el contenido necesita ser puesto a disposición de una manera segura y confiable. Un ejemplo es el intercambio de archivos desde un ordenador personal a través de internet, un usuario que tiene Apache instalado en su escritorio puede colocar archivos arbitrarios en la raíz de documentos del servidor y pueden ser compartidos.

Aunque el objetivo principal del servidor no es ser el más rápido, Apache tiene un rendimiento comparable a otros servidores web de "alto rendimiento". Se compromete con el funcionamiento de manera que reduce la latencia y aumenta el rendimiento en los momentos de más peticiones, así se asegura un tratamiento constante y fiable de peticiones dentro de periodos razonables.

2.1.5 Aplicaciones estáticas y dinámicas

Las aplicaciones web se dividen en dos grandes grupos; webs estáticas y webs dinámicas. A continuación, se analizarán las principales diferencias entre ambas, lo que permitirá entender la elección tomada para realizar esta aplicación web.

Una web estática presenta las siguientes características:

- Ausencia de movimiento y funcionalidades
- Absoluta opacidad a los deseos o búsquedas del visitante a la página
- Realizadas en XHTML o HTML
- Para modificar los contenidos de la página, es imprescindible acceder al servidor donde está alojada la página
- El usuario no tiene posibilidades de seleccionar, modificar u ordenar los contenidos o el diseño de la página a su gusto
- El proceso de actualización es lento, tedioso y esencialmente manual
- No se pueden utilizar funcionalidades tales como bases de datos, foros...

Por el contrario, una web dinámica tiene las siguientes características:

- Gran número de posibilidades en su diseño y desarrollo
- El visitante puede alterar el diseño, contenidos o presentación de la página a su gusto
- En su realización se pueden usar diferentes lenguajes y técnicas de programación

- El proceso de actualización es sencillo, sin necesidad de entrar en el servidor

- Permite un gran número de funcionalidades como bases de datos, foros...etc.

- Puede realizarse íntegramente con software libre

- Cuenta con un gran número de soluciones prediseñadas de libre disposición

Como idea principal podemos decir que si solo se requiere informar al público visitante sobre productos o servicios, una web estática sería la elección correcta, pero si se desea interactuar con nuestra web entonces se necesitaría una web dinámica.

Una web dinámica es mucho más costosa que una web estática en cuanto a la programación de la misma. No obstante, una web estática puede tener los mismos o más costes que una web dinámica, ya que, en algunas ocasiones, la complejidad reside en el grado de sofisticación del diseño.

No es suficiente tener un buen diseño para obtener una mayor rentabilidad del sitio web, también es necesario tener un contenido de calidad y actualizado. Hay que unir diseño y contenido y esto sólo se puede obtener con una web dinámica.

Como conclusión se puede decir que los dos tipos de web pueden ser una buena opción, depende de los objetivos planteados. Por tanto, nuestra aplicación será dinámica para poder cumplir los objetivos que nos hemos propuesto alcanzar.

2.2 Tecnologías aplicadas

Para la solución de este proyecto se ha utilizado el servidor independiente de software libre *'XAMPP[8]'.* Este servidor consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y PERL. El programa está liberado bajo la licencia GNU y actúa como servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Este servidor ha sido el elegido por su fácil instalación y porque permite testear el trabajo en el ordenador personal sin ningún acceso a internet.

En el proceso de construcción de la base de datos podemos distinguir entre dos fases; fase de diseño y de implementación. Para la fase de diseño se han utilizado los modelos de datos siguientes; el modelo E/R (Chen P.,1976) y el modelo relacional (Codd E., 1970). Por otro lado, para las fases de construcción, carga de la base de datos y recuperación de los datos en la aplicación, el lenguaje utilizado ha sido SQL.

La realización del modelo E/R o Entidad Interrelación ha servido para aclarar ideas sobre la situación y las relaciones de los conceptos antes de pasar a la realización definitiva del diseño lógico de la base de datos, además de que supone un nivel mayor de abstracción y por tanto, una forma más cómoda de trabajar al simplificar el problema. Representado en el apartado (3.2.1.1) de este documento.

El modelo relacional es el esquema lógico de la fase de diseño de la base de datos. Este modelo es propio del tipo de base de datos que se va a implementar, tipo relacional. El diseño lógico debe ser de este tipo para obtener así una implementación directa desde el esquema diseñado. Representado en el apartado (3.2.1.2) de este documento.

[8] *www.apachefriends.org/es/xampp.html*

En las tareas de construcción y carga de la base de datos, así como la recuperación de información de la misma, nos hemos centrado en un único lenguaje, teniendo en cuenta que debía ser compatible con el sistema gestor de bases de datos, en este caso es MySQL. La tecnología elegida es el lenguaje declarativo de bases de datos de alto nivel SQL. El lenguaje fue estandarizado por ANSI (Instituto Nacional Estadounidense de Estándares) y posteriormente aceptado por ISO (Organización Internacional para la Estandarización), este último es el organismo de estandarización internacional más importante.

2.2.1 Descripción del Sistema Gestor de Bases de Datos

MySQL es un Sistema Gestor de Bases de Datos relacionales, multiusuario, multihilo y desarrollado en ANSI C y C++. Por un lado se ofrece bajo la GNU/GPL para cualquier uso compatible con ésta, sin embargo, aquellas empresas que quieran incorporarlo para fines privados deben comprar a MySQL AB una licencia específica que les permita este uso.

MySQL es muy utilizado en plataformas Linux/Windows-Apache-MySQL-PHP/Perl/Python. Su popularidad como aplicación Web está muy ligada a PHP, que a menudo aparece como combinación con MySQL, ya que gracias a su rapidez es el gestor ideal para este tipo de aplicaciones. A continuación se listan algunas de sus características más destacadas:

- Soporte a multiplataforma
- Un amplio subconjunto de ANSI SQL 99, y varias extensiones
- Procedimientos almacenados
- Cursores
- Disparadores ("*trigger*")
- Soporte a Varchar
- Vistas actualizables
- Motores de almacenamiento independientes (MyISAM para lecturas rápidas, InnoDB para transacciones e integridad referencial)
- Soporte para SSL
- Permite sentencias SELECT anidadas
- Soporte completo para Unicode

En el paquete de aplicaciones elegido para desarrollar el proyecto (XAMPP) se incluyen módulos como Open SSL o phpMyAdmin. Esta última es una herramienta escrita en PHP que actúa de interfaz con el sistema gestor de bases de datos a través de aplicaciones web, que permite ejecutar sentencias SQL en el navegador web y realizar operaciones sobre bases de datos, administración, tablas e importación y exportación de datos.

Durante todo el desarrollo del proyecto se ha utilizado esta aplicación para interactuar con la base de datos. En la siguiente figura se muestra la apariencia de la herramienta con la base de datos del proyecto en pantalla:

Ilustración 2: phpMyAdmin

Este paquete de aplicaciones ha sido elegido por ser el más adecuado y adaptarse a las necesidades planteadas. Este servidor contiene todas las herramientas necesarias para la realización de las tareas que comprenden este proyecto.

Por comodidad a la hora de trabajar y para evitar imprevistos, se ha optado por una versión portable del servidor. Esta versión permite transportar toda la aplicación de manera sencilla en una memoria portátil USB, de este modo podemos trabajar en diferentes máquinas lo que facilita el trabajo del desarrollador.

El manual completo de instalación se encuentra en los anexos a este documento.

2.2.2 SQL

El lenguaje de consulta estructurado o SQL es un lenguaje declarativo de acceso a bases de datos relacionales que permite la especificación de diversos tipos de operaciones sobre éstas. Actualmente es uno de los lenguajes de bases de datos más usado, tanto es así que SQL es el estándar que deben seguir los sistemas gestores de bases de datos relacionales.

Está considerado un lenguaje de alto nivel gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, lo que permite una alta productividad en codificación y orientación a objetos. De este modo una única sentencia podría equivaler a uno o más programas que se utilicen en un lenguaje de bajo nivel.

Está compuesto por dos grandes sub-lenguajes con una temática bien diferenciada. Para la declaración de las estructuras y objetos de la base de datos se utilizará el Lenguaje de Definición de Datos (LDD) y para las fases de carga y consulta, SQL dispone del Lenguaje de Manipulación de Datos (LMD).

Para el lenguaje LDD existen cuatro operaciones básicas, todas ellas pueden actuar sobre una tabla, vista, índice, *trigger*, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte:

- CREATE: Este comando crea un objeto dentro de la base de datos.
- ALTER: Este comando permite modificar la estructura de un objeto.
- DROP: Este comando elimina un objeto de la base de datos. Se puede combinar con la sentencia ALTER.

- TRUNCATE: Este comando trunca toda el contenido de una tabla. La ventaja sobre el comando DROP, es que si se quiere borrar todo el contenido de la tabla, es mucho más rápido, especialmente si la tabla es muy grande. La desventaja es que TRUNCATE sólo sirve cuando se quiere eliminar absolutamente todos los registros, ya que no se permite la cláusula *Where*.

El lenguaje LMD es un lenguaje proporcionado por el sistema de gestión de bases de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, existen cuatro operaciones básicas:

- INSERT: Agrega uno o más registros a una, y sólo una, tabla en una base de datos relacional. Las cantidades de columnas y valores deben ser iguales. Si una columna no se especifica le será asignado el valor por omisión.
- UPDATE: Esta sentencia es utilizada para modificar los valores de un conjunto de registros existentes en una tabla
- DELETE: Borra uno o más registros existentes en una tabla
- SELECT: Selecciona uno o más registros de una tabla de la base de datos

2.2.3 Descripción de estándares web

Los estándares web es un término muy general utilizado para referirse a estándares y otras especificaciones que definen y describen aspectos de *la World Wide Consortium* (W3C). Cuando se describe que un sitio web cumple con ciertos estándares web, usualmente quiere decir que la página tiene partes de código HTML, CSS y Java Script válido o casi válido. La parte HTML debe cumplir también ciertas guías de accesibilidad y semántica.

Un sitio basado en estándares web mostrará una mayor consistencia visual. Gracias al uso de XHTML para el contenido y CSS para la apariencia, se puede transformar rápidamente un sitio web. Los documentos que separan apariencia de contenido usan menos código, además, CSS permite conseguir efectos que antes requerían el uso de JavaScript e imágenes, por lo que los sitios basados en estándares utilizan menos ancho de banda y se muestran más rápido a los usuarios.

Utilizando los estándares web, el sitio web es más fácil de mantener y actualizar, el código es más simple y más accesible, permitiendo a personas con discapacidades utilizar su contenido.

2.2.4 HTML

HTML son las siglas de *HyperText Markup Language* (Lenguaje de Marcas de Hipertexto), es el utilizado para la construcción de páginas web. Su uso está destinado a describir la estructura y contenido de las sitios web en forma de texto, así como para complementar el texto con objetos tales como imágenes.

Este lenguaje se construye mediante etiquetas que delimitan las secciones de la web y alojan la información. Normalmente suele estar implementado junto con hojas de estilo llamadas CSS que ayudan a definir los aspectos visuales de la web. HTML puede incluir scripts (JavaScript, PHP) que afectan al comportamiento de navegadores web y otros procesadores de HTML.

Puede ser creado y editado por cualquier editor de textos básico, como Gedit en Linux o Notepad++ en Windows, simplemente hay que tener en cuenta que la extensión del documento ha de ser *.htm o *.html. Dentro de estos archivos se implementará el código HTML el cual consta de varias etiquetas básicas:

- **<head>** : Define la cabecera del documento HTML, esta cabecera suele contener información sobre el documento que no se muestra directamente al usuario, como puede ser el título de la ventana del navegador.

- **<title>** : Define el título de la página que aparecerá encima de la ventana del sitio web.

- **<html>** : Define el inicio de un documento HTML, de esta forma se le indica al navegador que el texto que le sigue debe ser interpretado como código HTML.

- **<script>** : Introduce un script en una web, o se llama a uno mediante el atributo "*src= 'url del script'*".

- **<link>** : Para vincular el sitio a hojas de estilo o iconos.

- **<style>** : Coloca el sitio interno de la página, ya sea usando CSS u otros lenguajes similares.

- **<a>** : Para crear un enlace es necesario utilizar esta etiqueta de ancla junto al atributo href, que establecerá la dirección URL a la que apunta el enlace.

- **<body>** : Define el cuerpo o contenido principal del documento. Esta es la parte del documento que se muestra en el navegador. Dentro del cuerpo se pueden definir numerosas etiquetas, las más utilizadas son:

    - **<p>** : Define un párrafo de texto en la web.

    - **
** : Introduce un salto de línea

    - **<h1>** : Indica que se va a escribir una cabecera para una sección, el número indica el tamaño de la fuente.

    - **<td>** : Define tablas en la página.

    - **<li>, <lu>** : Definen listas de elementos, ya sean ordenadas o no.

2.2.5 Hoja de estilos CSS

CSS o lo que es lo mismo, hojas de estilo en cascada, es un lenguaje utilizado para definir la presentación de un documento escrito en HTML o XML. El *W3C* es el encargado de formular las especificaciones de las hojas de estilo que servirán de estándar para los navegadores.

La idea que se mantiene para el desarrollo de las hojas de estilo es separar la estructura del documento de su presentación, lo que hace más fácil la modificación de los estilos en caso de ser necesario.

Cuando se utiliza CSS, las etiquetas no deben proporcionar información acerca de la visualización del documento, sino marcar la estructura. La información del estilo, separada en una hoja de estilo, especifica como se ha de mostrar una etiqueta; color, fuente, alineación del texto o tamaño.

La introducción de CSS ha permitido en muchos casos reemplazar el uso de tablas, que era la técnica que se utilizaba antes para conseguir el mismo fin. Sin embargo, CSS todavía no permite la misma versatilidad pues las diferencias entre los distintos navegadores dificultan la tarea. Se espera que futuros desarrollos en CSS3 resuelvan esta deficiencia y hagan de CSS un lenguaje más apto para describir la estructura espacial de una página. Como ventajas caben destacar:

- Los navegadores permiten a los usuarios especificar la hoja de estilo local que será aplicada en un sitio web, con lo que se obtiene una mayor accesibilidad.

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza la actuación del mismo de forma considerable.

- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

2.2.6 PHP

El lenguaje PHP es un lenguaje de programación diseñado y orientado a la creación de páginas dinámicas. Es utilizado principalmente en interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otro tipo de aplicaciones en las que se encuentran aquellas con interfaz gráfica.

La mayor fuerza de PHP reside en que es un lenguaje preparado para soportar accesos a muchos tipos de bases de datos como Oracle, ODBC, DB2, SQLServer...etc., y enfocando nuestro caso, MySQL. Lo que hace diferente a PHP es que el código que se deba ejecutar se ejecuta siempre en el servidor, de este modo el cliente solo recibe los resultados de la ejecución y le es imposible acceder al código que generó la página.

Las conexiones de PHP a bases de datos son enlaces SQL que no se cierran cuando termina la ejecución del script. El comportamiento de estas conexiones consiste en que al invocar una conexión, PHP comprueba si ya existe una conexión de este mismo tipo o es una conexión nueva. En el caso de que exista se procede a su uso y en caso que no exista la conexión se crea. Dos conexiones se consideran iguales cuando están realizadas al mismo servidor y con un mismo usuario y contraseña.

El código PHP se encuentra inmerso entre el código HTML y Java Script distinguido entre etiquetas:

<? php
código...
..... ?>

2.2.7 JavaScript

JavaScript es un lenguaje basado en objetos y guiado por eventos, utilizado para acceder a objetos en aplicaciones. Se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. Es un lenguaje influenciado de múltiples lenguajes y fue diseñado con el objetivo de tener una sintaxis similar a la de Java, aunque más sencilla para facilitar el uso a programadores principiantes. Todos los navegadores actuales interpretan el código de JavaScript integrado dentro de las páginas web.

Entre las acciones típicas que se pueden realizar en JavaScript tenemos dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien color o cualquier otro dinamismo. Por el otro, JavaScript nos permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que podemos crear páginas interactivas con programas como calculadoras, agendas o tablas de cálculo.

Este lenguaje se puede incluir en cualquier documento y es compatible con cualquier sistema operativo. La mejor manera es incluir JavaScript como un archivo externo, tanto por cuestiones de accesibilidad como por la velocidad en la navegación, este archivo será del tipo *.js y para utilizarlo se debe escribir en el documento HTML:

<script type="text/javascript" src="[url del fichero js]"></script>

También es posible incrustar el código en el documento con la etiqueta <script>:

<script type="text/javascript"><!--//codigo JavaScript--></script>

2.2.8 Librerías JPGraph

Al utilizar las herramientas de Google Maps nos hemos encontrado alguna dificultad para realizar los objetivos de este proyecto. La idea principal era representar, en las pestañas de los mapas de Google, gráficos estadísticos sobre los datos almacenados, estos gráficos también se iban a realizar con herramientas de Google puesto que resultaban más vistosos y óptimos, pero este trabajo ha sido imposible llevarlo a cabo pues las pestañas solo admiten código HTML, por tanto se limitaba el contenido de las pestañas a imágenes, texto, tablas o botones.

Como alternativa se han utilizado las librerías JPGraph, estas librerías gratuitas sirven para crear imágenes con todo tipo de gráficas, dinámicamente desde páginas PHP. Incluye una serie de clases en código orientado a objetos que ofrecen una solución muy interesante para la generación de las imágenes con las gráficas, de manera que solo hay que preocuparse de cargar los datos a representar y escoger el tipo de gráfica que se desea visualizar.

El modo de trabajo para usar esta librería es muy simple, se trata de crear una imagen con la etiqueta <img> de HTML, en cuyo atributo src se coloca la ruta hacia el script PHP que se encargará de generar la gráfica, resultando muy fácil integrar el gráfico en el diseño web.

En el archivo PHP que generará la gráfica tendremos que incluir las librerías apropiadas para el tipo de gráfica que se va a utilizar, también habrá que instanciar el objeto JPGraph correspondiente, cargar los datos a visualizar y llamar a los métodos adecuados para mostrar la imagen.

2.2.9 Ajax

Ajax es el acrónimo de *Asynchronous JavaScript And XML*. Es una técnica de desarrollo web para crear aplicaciones interactivas. El concepto es cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en background, los datos que son usados para actualizar los gráficos de la página.

Es una tecnología asíncrona, lo datos se obtienen de la base de datos y se cargan en segundo plano sin interferir en el uso de la aplicación para el usuario. En este tipo de aplicaciones se introduce un intermediario, un motor Ajax, entre el usuario y el servidor. En vez de cargar una página web, al inicio de la sesión, el navegador carga al motor Ajax para que sea el responsable de renderizar la interfaz y comunicarse con el servidor en nombre del usuario.

El motor Ajax permite que la interacción del usuario con la aplicación suceda asíncronamente, de esta forma el usuario nunca estará mirando una ventana en blanco del navegador.

La única manera de realizar el proyecto sin Ajax sería mediante la creación de un formulario, cuando el usuario envíe los datos del formulario es cuando se realizaría la conexión con la base de datos y se mostraría por pantalla la página que devuelve el servidor. Con este método se recargaría la página bien saltando a una nueva o a la misma, el usuario tendría que soportar la espera de cada recarga y este proceso podría resultar lento puesto que la información HTML se descarga por duplicado.

Por todos estos motivos hemos utilizado Ajax y por tanto un código JavaScript es el utilizado para crear un objeto XMLHttpRequest al enviar el formulario, los datos se envían al servidor pero no se recarga la página, posteriormente el servidor responde la petición y una función JavaScript es la que valora la respuesta del servidor, si la respuesta es válida se imprime el texto al usuario. Este método es bastante más rápido porque el código HTML de la página de confirmación del formulario ya no se tiene que descargar.

A continuación vamos a citar las diez razones por la que considerar el uso de Ajax que vienen comentadas en numerosos sitios web dedicados a desarrolladores[9]

1. **Basado en los estándares abiertos**: está formado por las tecnologías más usadas y que soportan los navegadores más utilizados de internet.
2. **Usabilidad**: Elimina tener que refrescar el navegador cuando se realiza una petición de datos al servidor.
3. **Válido en cualquier plataforma y navegador**: Los navegadores basados en mozilla[10] y firefox son los más sencillos para programar aplicaciones web Ajax pero ahora es posible que esas aplicaciones funcionen en los navegadores más modernos.
4. **Beneficia las aplicaciones Web**: Cada vez se utilizan más las aplicaciones web puesto que requieren un menor coste de creación y facilitan el soporte y mantenimiento. Ajax ayuda a estas aplicaciones a mejorar y conseguir un mejor resultado para el usuario.
5. **No es difícil su utilización**: Ajax está basada en los estándares que se utilizan y por tanto los desarrolladores no requieren de un gran esfuerzo para su aprendizaje.
6. **Compatible con Flash**: Existen ventajas y desventajas para ambas tecnologías por lo que el uso combinado ofrece buenas alternativas.

[9] *http://www.webnova.com.ar/articulo.php?recurso=566*
[10] *www.mozilla-europe.org/es/firefox*

7. **Adoptado por los "gordos" de la tecnología web**: El mercado acepta y valida el uso de esta tecnología. Grandes de la industria como Google, Yahoo[11], Amazon[12], Microsoft la utilizan.

8. **Web 2.0**: Una de las claves de la web 2.0 es usar la red como plataforma de desarrollo de aplicaciones siendo muy importante la iteración de los usuarios.

9. **Es independiente del tipo de tecnología de servidor que se utilice**: Ajax es compatible con cualquier tipo de servidor estándar y lenguaje de programación web.

10. **Mejora la estética de la web**: Con Ajax se pueden realizar aplicaciones que podrían ser de escritorio ya que permite combinar la imaginación de los desarrolladores con la usabilidad.

[11] *es.yahoo.com/*

[12] *www.amazon.com/*

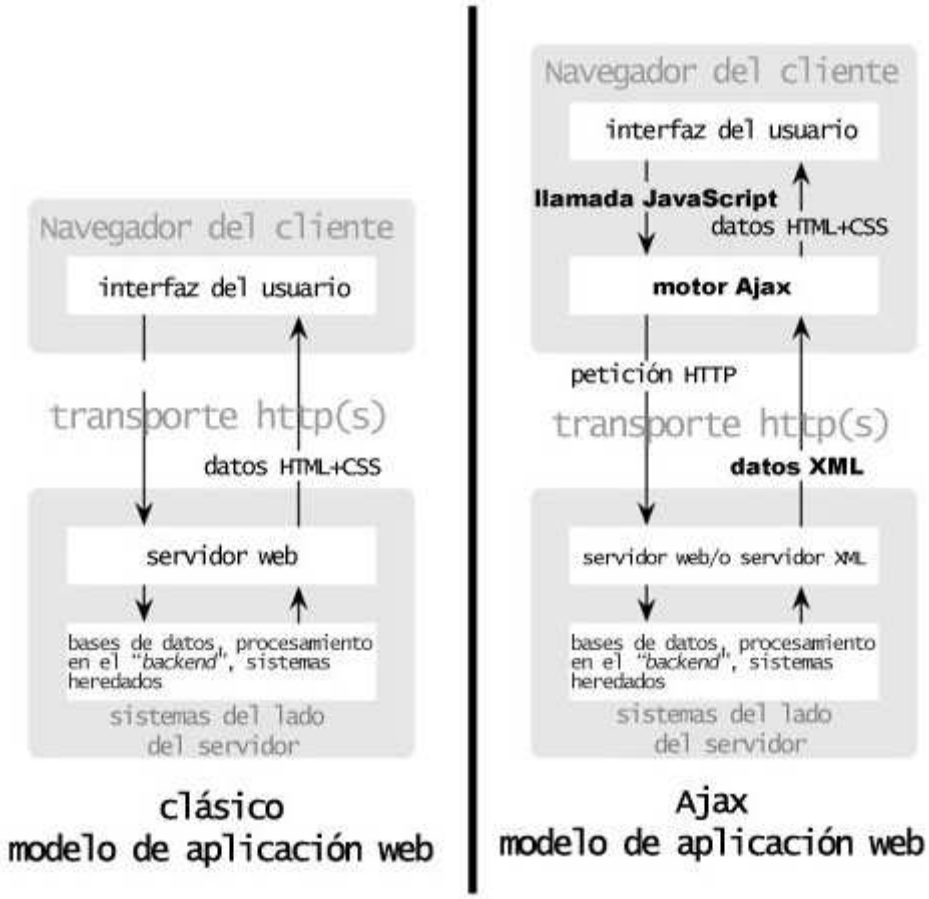


Ilustración 3: Comparativa entre Ajax y el modelo tradicional[13]

La figura anterior muestra la comparativa entre el modelo tradicional para el desarrollo de aplicaciones web y el modelo utilizado para Ajax. Se puede observar que la gran diferencia reside en el motor Ajax que actúa de intermediario entre el cliente y el servidor.

[13] *www.maestrosdelweb.com*

2.2.10 Google Maps

Google Maps[14] es un servicio que ofrece mapas del mundo, en principio como complemento y ayuda al usuario que realiza búsquedas en 'Google Local'. A partir de 2005 comenzó a ofrecer imágenes vía satélite y de esta manera se permitió ver fotografías aéreas de todo el planeta de mayor o menor resolución dependiendo si se trata o no de importantes núcleos urbanos.

Los mapas de Google es un servicio extraordinario, existen tutoriales que la compañía ha decidido lanzar y ponérselo fácil a los desarrolladores 'API de Google Maps' que guían el camino para integrarlo con herramientas de nuestro sitio web. De esta manera, Google proporciona una serie de procedimientos bien documentados para que, a través de código JavaScript en nuestras páginas web, podamos comunicarnos con sus servidores y extraer los datos de los mapas.

Para poder utilizar esta API es necesario solicitar una clave e indicar en qué URL vamos a alojar la aplicación. Google solamente facilita 50 mil consultas a sus servidores por día, y anuncia que esta API será actualizada periódicamente y se reserva el derecho de insertar publicidad dentro de los mapas.

El API de Google Maps es un servicio gratuito que permite insertar Google Maps en páginas web propias con JavaScript. Las Api proporcionan diversas utilidades para manipular y añadir contenido al mapa mediante diversos servicios, permitiendo crear sólidas aplicaciones de mapas.

[14] *maps.google.es*

Capítulo 3

Análisis, Diseño e Implementación

En este capítulo se van a desarrollar las diferentes tareas realizadas en las fases de análisis, diseño e implementación diferenciando en cada apartado entre la base de datos y la plataforma web.

3.1 Análisis

3.1.1 Requisitos del sistema

A continuación, se definen cada uno de los campos de las tablas en las que se describen los requisitos software.

- **ID**: Identificador univoco del requisitos. Debe seguir la siguiente nomenclatura:

  - RSF_XX: Requisitos Software Funcionales

  -RSNF_XX: Requisitos Software no Funcionales

  - RSU_XX: Requisitos Software de Usabilidad

  Siendo XX números comprendidos entre 01 y 99 que se incrementarán en una unidad consecutiva con cada nuevo requisito.

- **Tipo**: Pueden ser funcionales, de interfaz, de usabilidad o de mantenimiento.

- **Patrón(es)**: Enumera los patrones en los que se ha basado para realizar el requisito. Este campo solamente estará presente en los requisitos funcionales y de usabilidad, si procede.

- **Descripción**: Explicación del requisito.

- **Importancia**: Define la importancia de la funcionalidad que aporta el requisito al proyecto. El dominio de este campo puede tomar los valores "Vital", "Importante" o "Aconsejable".

- **Justificación**: Motivo por el cual se ha decidido incluir el requisito

- **Criterio de cumplimiento**: Método para comprobar que el requisito se ha cumplido

❖ Requisitos Funcionales:

ID	RSF_01
Tipo	Funcional
Patrón(es)	ME1 *(Hierarchical Organization)*, BE1 *(Collection Center)*, BE2 *(Node as a Single Unit)*, BN2 *(Site Maps)*, AE1 *(User centred structure)*
Descripción	Al tratar una cantidad ingente de información se establece un árbol jerárquico con los conceptos extraídos para establecer niveles de importancia y relaciones para poder así clasificarla y representarla de manera lógica.
Importancia	Vital
Justificación	Sin este requisito no se podrían establecer módulos para los distintos apartados y la información estaría reflejada sin ningún orden.
Criterio de cumplimiento	Observando las pestañas y los distintos caminos a elegir en la aplicación, que la información está organizada.

Tabla 2: Requisito RSF_01

ID	RSF_02
Tipo	Funcional
Patrón(es)	D2 *(Content modules)*, B8 *(Category pages)*, K3 *(Tab Rows)*
Descripción	Se realiza una división del contenido en dos grandes grupos para las distintas secciones de manera que se consiga evitar que la información aparezca mezclada y poco clara
Importancia	Vital
Justificación	Se realiza para separar contenidos y facilitar el entendimiento del usuario sobre los distintos contenidos
Criterio de cumplimiento	La página principal distingue los dos grandes módulos de información a los que el usuario puede acceder

Tabla 3: Requisito RSF_02

ID	RSF_03
Tipo	Funcional
Patrón(es)	E3 *(Fair information practices*), E4 *(Privacy Policy)*
Descripción	En todas las secciones se podrá consultar la política de privacidad de datos que mantiene la aplicación para así cumplir la Ley Orgánica de Protección de Datos además de dar fiabilidad al usuario
Importancia	Importante
Justificación	Cumplimiento de la legislatura en vigor así como para proporcionar más fiabilidad al usuario
Criterio de cumplimiento	La parte inferior de cada una de las páginas tiene un enlace donde el usuario podrá acceder a un texto en el que se encuentran definidos los pasos a seguir por la aplicación para el tratamiento de los datos

Tabla 4: Requisito RSF_03

ID	RSF_04
Tipo	Funcional
Descripción	Deberá existir una sección en la aplicación llamada "Contacto" donde aparezcan los nombres de los administradores de la aplicación en caso de necesitar realizar alguna consulta
Importancia	Importante
Justificación	La aplicación debe tener unos contactos que se encarguen del mantenimiento en caso de ser necesario
Criterio de cumplimiento	La parte inferior de cada una de las páginas tiene un enlace donde el usuario podrá acceder a un texto con la información

Tabla 5: Requisito RSF_04

ID	RSF_05
Tipo	Funcional
Descripción	Deberá existir una sección en la aplicación llamada "Política de privacidad" donde aparezcan las políticas usadas por la organización para el manejo de datos personales
Importancia	Importante
Justificación	La aplicación debe seguir una política de privacidad de datos que cumpla la legislatura vigente
Criterio de cumplimiento	La parte inferior de cada una de las páginas tiene un enlace donde el usuario podrá acceder a un texto con la información

Tabla 6: Requisito RSF_05

ID	RSF_06
Tipo	Funcional
Descripción	Para comenzar a usar la aplicación el usuario deberá seleccionar el tipo de zona a la que se desea acceder, rural o urbana
Importancia	Importante
Justificación	La aplicación realiza una clara distinción entre los dos tipos de zonas
Criterio de cumplimiento	En la página principal solo existe los enlaces para los tipos de zonas, no se puede acceder a una zona concreta sin haber elegido el tipo en la página principal

Tabla 7: Requisito RSF_06

ID	RSF_07
Tipo	Funcional
Descripción	Dentro de un tipo de zona el usuario podrá moverse entre las distintas zonas
Importancia	Importante
Justificación	La aplicación realiza una clara distinción entre los dos tipos de zonas
Criterio de cumplimiento	Una vez seleccionada un tipo de zona se puede navegar entre los distintos tipos de manera rápida

Tabla 8: Requisito RSF_07

ID	RSF_08
Tipo	Funcional
Descripción	El administrador podrá modificar la información desde el gestor de bases de datos
Importancia	Vital
Justificación	La información debe estar controlada y los usuarios no tendrán acceso para no almacenar datos erróneos
Criterio de cumplimiento	La base de datos tiene una contraseña que solo conoce el administrador

Tabla 9: Requisito RSF_08

ID	RSF_09
Tipo	Funcional
Descripción	El usuario podrá comenzar la simulación de los datos en el momento deseado
Importancia	Importante
Justificación	Al no estar la aplicación en un servidor, se debe poder elegir cuando iniciar la recogida de los datos
Criterio de cumplimiento	Existe un botón para iniciar la simulación de cada zona

Tabla 10: Requisito RSF_09

ID	RSF_10
Tipo	Funcional
Descripción	El usuario podrá terminar la simulación de los datos en el momento deseado
Importancia	Importante
Justificación	De este modo, el usuario puede parar una simulación de un sensor en el momento que se desee, para evitar malgastar recursos
Criterio de cumplimiento	En el momento que se cierra la pestaña del mapa se acaba la simulación de ese sensor

Tabla 11: Requisito RSF_10

ID	RSF_11
Tipo	Funcional
Descripción	El usuario rellenará el campo para la simulación de una lista desplegable
Importancia	Importante
Justificación	De este modo, el usuario no puede iniciar una simulación de un sensor que no exista
Criterio de cumplimiento	El campo a rellenar tiene una lista desplegable cuyos valores son los únicos aceptados

Tabla 12: Requisito RSF_11

ID	RSF_12
Tipo	Funcional
Descripción	Los mapas tendrán la posibilidad de agrandarlos en una zona seleccionada
Importancia	Importante
Justificación	El usuario podrá hacer zoom para ver con mas detalle la zona
Criterio de cumplimiento	Los mapas constan de unos botones en la parte izquierda para agrandar o disminuir el nivel de detalle

Tabla 13: Requisito RSF_12

ID	RSF_13
Tipo	Funcional
Descripción	Los mapas tendrán la posibilidad de cambiar el tipo de vista seleccionada
Importancia	Importante
Justificación	El usuario podrá modificar el tipo de vista en la que se desea ver el mapa
Criterio de cumplimiento	Los mapas constan de unos botones en la parte superior derecha para modificar la vista del mapa

Tabla 14: Requisito RSF_13

ID	RSF_14
Tipo	Funcional
Descripción	Se deberá implementar una interfaz de usuario que sea útil, intuitiva y consistente.
Importancia	Vital
Justificación	El usuario deberá manejar la aplicación de forma fácil y sin necesitar formación previa
Criterio de cumplimiento	Los títulos de la aplicación son bastante intuitivos para el usuario

Tabla 15: Requisito RSF_14

ID	RSF_15
Tipo	Funcional
Descripción	La aplicación presentará una barra de pie de página o *footer bar*
Importancia	Importante
Justificación	La aplicación tendrá un pie de página con enlaces que el usuario podrá acceder en cualquier página
Criterio de cumplimiento	La aplicación tiene en todas sus seccione un pie de página

Tabla 16: Requisito RSF_15

ID	RSF_16
Tipo	Funcional
Descripción	La aplicación presentará una portada para dar la bienvenida al usuario
Importancia	Importante
Justificación	La aplicación tendrá una página de inicio para situar al usuario
Criterio de cumplimiento	La aplicación tiene una página principal

Tabla 17: Requisito RSF_16

ID	RSF_17
Tipo	Funcional
Descripción	Únicamente el administrador de la base de datos podrá modificar los datos
Importancia	Importante
Justificación	Se debe tener un control y verificación de los datos
Criterio de cumplimiento	La base de datos tiene una clave de acceso

Tabla 18: Requisito RSF_17

- Requisitos No Funcionales:

ID	RSNF_01
Tipo	No funcional
Descripción	El cliente debe ser capaz de acceder a la aplicación 24 horas al día, siete días a la semana.
Importancia	Vital
Justificación	La aplicación debe estar siempre disponible por si ocurre una emergencia
Criterio de cumplimiento	Comprobar en distintos horarios que la aplicación está disponible

Tabla 19: Requisito RSNF_01

ID	RSNF_02
Tipo	No funcional
Descripción	Será posible el cambio de estilo gracias al uso de CSS
Importancia	Importante
Justificación	Puede ser necesario un cambio de estilo global y debe ser fácilmente realizable
Criterio de cumplimiento	Probar un cambio de estilo

Tabla 20: Requisito RSNF_02

ID	RSNF_03
Tipo	No funcional
Descripción	En cada sección, el cliente recibe la información gráficamente mediante los enlaces que se encuentran en el interior de cada mapa
Importancia	Vital
Justificación	De esta forma se permite un acceso más rápido
Criterio de cumplimiento	Comprobar que la información de los mapas se muestra de manera gráfica

Tabla 21: Requisito RSNF_03

❖ Requisito de Usabilidad:

ID	RSU_01
Tipo	Usabilidad
Patrón(es)	K10 *(Obvious links)*, BI1 *(Action buttons)*, AI1 *(Interaction)*
Descripción	Todos los enlaces del sitio web utilizados para relacionar los distintos módulos de la aplicación estarán representados de una manera especial para que sean fácilmente distinguidos por los usuarios
Importancia	Vital
Justificación	Es importante no confundir al usuario
Criterio de cumplimiento	Todos los enlaces están resaltados del texto

Tabla 22: Requisito RSU_01

ID	RSU_02
Tipo	Usabilidad
Patrón(es)	MI1 *(Information on demand)*
Descripción	La información se representa de manera que el usuario pueda observarla en su totalidad, sin la necesidad de utilizar las barras de desplazamiento
Importancia	Importante
Justificación	Se muestra al usuario la información de manera clara y concisa
Criterio de cumplimiento	No hay barras de desplazamiento en las páginas principales

Tabla 23: Requisito RSU_02

ID	RSU_03
Tipo	Usabilidad
Patrón(es)	BN1 *(Location Bread Crumbs)*, BP1 *(Navigation Bar)*
Descripción	La información queda representada de forma que los usuarios puedan unir las principales categorías de información de una manera secuencial y ordenada
Importancia	Vital
Justificación	Se recuerda al usuario los pasos previos hasta alcanzar la situación actual, de manera que favorece la interactuación del usuario
Criterio de cumplimiento	Se puede observar la situación del usuario dentro de la jerarquía

Tabla 24: Requisito RSU_03

ID	RSU_04
Tipo	Usabilidad
Patrón(es)	AN1 *(Multiple Ways to Navigate)*, BP1 *(Navigation Bar)*, MN2 *(Guided tour Navigation)*, BN1 *(Location Bread Crumbs)*
Descripción	Todas las páginas de la aplicación contienen una barra de navegación que permiten al usuario moverse libremente de forma rápida y sencilla por los distintos servicios de la misma.
Importancia	Vital
Justificación	Facilita al usuario la localización dentro de la aplicación
Criterio de cumplimiento	Comprobar que existe la barra de navegación

Tabla 25: Requisito RSU_04

ID	RSU_05
Tipo	No funcional
Patrón(es)	MI2 *(Process Feedback)*
Descripción	En el proceso en el que la aplicación está cargando los distintos mapas que debe mostrar al usuario, se refleja un mensaje para proporcionar información en todo momento de los procesos internos que se están realizando.
Importancia	Importante
Justificación	Tanto en búsquedas, como a la hora de mostrar mucho contenido informativo referente a una base de datos, y en procesos de gestión de archivos, mostraremos mensajes al usuario sobre el estado del proceso en cuestión para que no abandone la página pensando que ha habido un error en la aplicación.
Criterio de cumplimiento	Mensaje con información sobre el estado de la página

Tabla 26: Requisito RSU_05

ID	RSU_06
Tipo	Usabilidad
Patrón(es)	L6 *(Fast-Loading Contents)*
Descripción	Se minimizará el tiempo máximo posible de la carga de la aplicación
Importancia	Vital
Justificación	Se optimizará el código para que la carga se realice en el menor tiempo posible
Criterio de cumplimiento	Código optimizado en funciones independientes y claramente diferenciables

Tabla 27: Requisito RSU_06

ID	RSU_07
Tipo	Usabilidad
Patrón(es)	AP1 *(Aesthetics)*, B9 *(Site accessibility)*
Descripción	La página tendrá una gama de colores acordes con los temas que abarca y que faciliten la lectura
Importancia	Importante
Justificación	Se utilizan unos colores que resulten agradables para el usuario y ayuden a la navegación a personas con algún tipo de discapacidad visual
Criterio de cumplimiento	Evitar colores combinar colores fuertes y demasiado luminosos

Tabla 28: Requisito RSU_07

ID	RSU_08
Tipo	Usabilidad
Patrón(es)	B3 *(Hierarchical organization)*
Descripción	La aplicación tendrá una organización jerárquica
Importancia	Vital
Justificación	Las jerarquías ayudan a tener una visión global del sistema
Criterio de cumplimiento	Observar que en algunas secciones hay jerarquías

Tabla 29: Requisito RSU_08

ID	RSU_09
Tipo	Usabilidad
Patrón(es)	D1 *(Page templates)*
Descripción	Se debe usar una misma plantilla para toda la aplicación web
Importancia	Importante
Justificación	Proporciona uniformidad y fácil uso para el usuario
Criterio de cumplimiento	Comprobar que todas las páginas siguen el mismo estilo

Tabla 30: Requisito RSU_09

ID	RSU_10
Tipo	Usabilidad
Patrón(es)	E5 *(About us)*
Descripción	Se dispone de un apartado donde los desarrolladores quedan identificados
Importancia	Importante
Justificación	De esta manera los usuarios tendrán información sobre los creadores de la aplicación web
Criterio de cumplimiento	Comprobar que en el apartado Contacto hay información sobre los creadores

Tabla 31: Requisito RU_10

ID	RSU_11
Tipo	Usabilidad
Patrón(es)	K9 *(Describe Langer links names)*
Descripción	Utilizar nombres descriptivos en los enlaces o hipervínculos
Importancia	Importante
Justificación	De esta manera los usuarios tendrán información sobre los a dónde le llevará dicho enlace
Criterio de cumplimiento	Comprobar que los enlaces son entendibles

Tabla 32: Requisito RU_11

ID	RSU_12
Tipo	Usabilidad
Patrón(es)	K11 *(Familiar language)*
Descripción	El lenguaje que se utiliza se comprensible para el usuario al que va destinada la aplicación
Importancia	Importante
Justificación	Esto ayuda que los usuarios comprendan la aplicación fácilmente
Criterio de cumplimiento	Comprobar que el lenguaje utilizado es comprensible

Tabla 33: Requisito RU_12

3.1.2 Casos de uso

Un caso de uso es una técnica usada para realizar la captura de requisitos de un sistema informático. Cada caso de uso muestra los escenarios que indican cómo se interactúa entre el sistema y los usuarios del mismo. En este caso tenemos dos tipos de usuarios diferenciados:

- Personal: Aquellas personas que pertenecen a la organización en la que se utilice esta aplicación. Personas expertas en los parámetros que midan las redes de sensores implantadas y que accedan a la aplicación para interpretar los valores obtenidos.

- Administrador del sistema: Aquella con los permisos necesarios para llevar a cabo las funciones administrativas y con acceso a la base de datos para sus posibles modificaciones.

3.1.2.1 Descripción gráfica

En este apartado se muestran los diagramas de casos de uso representativos de las principales funcionalidades de los usuarios de la aplicación.

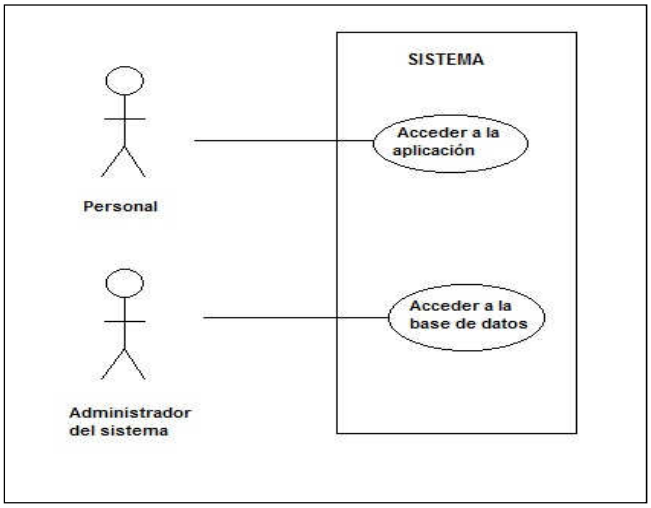


Ilustración 4: Caso de Uso nivel 0

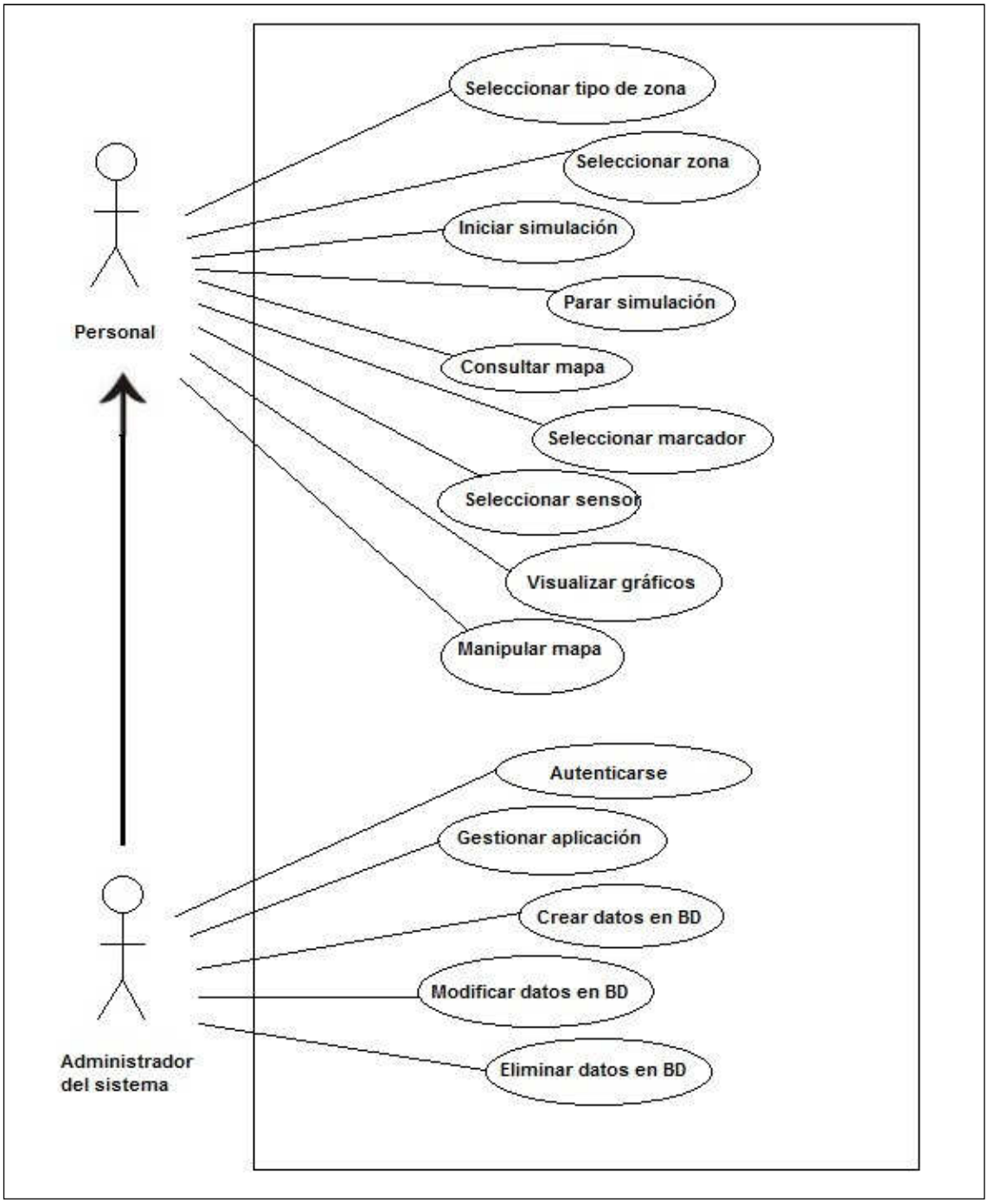


Ilustración 5: Caso de uso nivel 1

3.1.2.2 Descripción textual

Para cada uno de los casos de uso identificados en el apartado anterior se realizará una descripción textual para aclarar las funcionalidades. Los campos que describen los casos de uso son los siguientes:

- Identificador: Nombre de manera univoca un caso de uso. La nomenclatura a seguir será CU-XX siendo XX números comprendidos entre 0 y 9 que se irán incrementando consecutivamente.

- Nombre: Descripción breve del caso de uso.

* Tanto el campo *Identificador* como *Nombre* no estarán identificados en la tabla como campos sino que formarán parte del título de las mismas.

- Actor: Agente externo que interacciona con el sistema.

- Objetivo: Breve explicación del caso de uso.

- Precondiciones: Condiciones que deben cumplirse para poder ejecutar el caso de uso.

- Postcondiciones: Condiciones que se producen tras la ejecución del caso de uso.

- Escenario básico: Interacción más típica entre actor y sistema detallando la información y los cambios observados en el sistema.

- Escenario alternativo: Ejecución del caso de uso con condiciones de error o caminos de decisión distintos al básico.

CU-01	Seleccionar tipo de zona
Actor	Personal
Objetivo	Acceder al menú con las distintas zonas a elegir de un mismo tipo
Precondiciones	El usuario debe abrir la aplicación
Postcondiciones	El usuario se encuentra en el menú específico de un tipo de zona
Escenario básico	1. Se accede a la página principal de la aplicación 2. Se elige una de las imágenes de la portada que referencian los tipos de zonas
Escenario alternativo	2b. Se selecciona un enlace informativo del pie de página

Tabla 34: CU-01

CU-02	Seleccionar zona
Actor	Personal
Objetivo	Se accede a la página con la información de la zona concreta que se desea observar
Precondiciones	El personal ha seleccionado un tipo de zona
Postcondiciones	El personal puede visualizar el mapa, gráfico e información de la zona
Escenario básico	1. Se accede a la página principal de la aplicación 2. Se selecciona un tipo de zona 3. Se selecciona la zona concreta de las que aparezcan en el menú
Escenario alternativo	3b. Se selecciona un enlace informativo del pie de página

Tabla 35: CU-02

CU-03	Iniciar simulación
Actor	Personal
Objetivo	Se inicia la recogida de datos automática para la BD
Precondiciones	El servidor está conectado y abrir la página de simulación
Postcondiciones	Los datos generados de forma automáticas son almacenados en la BD
Escenario básico	1. Se abre la página de la simulación 2. Se elige el sensor del que se quieren obtener los datos 3. Se pulsa el botón iniciar
Escenario alternativo	2b. No se selecciona ningún sensor 3b. No se pulsa el botón de iniciar

Tabla 36: CU-03

CU-04	Parar simulación
Actor	Personal
Objetivo	Dejar de leer y almacenar datos del sensor
Precondiciones	Haber iniciado la simulación
Postcondiciones	La simulación queda finalizada
Escenario básico	1.Se visualiza el gráfico que se está actualizando por la simulación 2. Se cierra la ventana del gráfico
Escenario alternativo	No procede

Tabla 37: CU-04

CU-05	Consultar mapa
Actor	Personal
Objetivo	El personal visualiza el mapa de la zona seleccionada
Precondiciones	El personal ha seleccionado una zona para visualizar
Postcondiciones	El personal visualiza el mapa
Escenario básico	1. Se selecciona un tipo de zona 2. Se selecciona una zona concreta 3. Se visualiza el mapa de la zona
Escenario alternativo	2b. No se selecciona ninguna zona para visualizar 3b. No existe acceso a internet para visualizar los mapas

Tabla 38: CU-05

CU-06	Seleccionar marcador
Actor	Personal
Objetivo	Se selecciona un marcador en el que existen uno o mas sensores a visualizar
Precondiciones	Se visualiza el mapa de una zona concreta
Postcondiciones	El marcador queda seleccionado y se abre una pestaña con los gráficos de cada uno de los sensores de los que consta
Escenario básico	1. Se selecciona un tipo de zona 2. Se selecciona una zona concreta 3. Se visualiza el mapa de la zona 4. Se selecciona uno de los marcadores que aparecen en el mapa
Escenario alternativo	No procede

Tabla 39: CU-06

CU-07	Seleccionar sensor
Actor	Personal
Objetivo	Se selecciona un sensor del que se va a visualizar el gráfico
Precondiciones	Se selecciona un marcador de una zona en concreto
Postcondiciones	Se visualizan los datos referentes a ese sensor
Escenario básico	1. Se selecciona el tipo de zona 2. Se selecciona una zona concreta 3. Se selecciona un marcador del mapa de la zona 4. Se selecciona una de las pestañas que corresponde al sensor deseado
Escenario alternativo	No procede

Tabla 40: CU-07

CU-08	Visualizar gráficos
Actor	Personal
Objetivo	Visualizar los gráficos de un sensor seleccionado
Precondiciones	Se inicia la simulación de datos para ese sensor
Postcondiciones	El gráfico se va recargando con datos actualizados cada diez segundos
Escenario básico	1. Se selecciona el tipo de zona 2. Se selecciona una zona concreta 3. Se selecciona un marcador del mapa de la zona 4. Se selecciona una de las pestañas que corresponde al sensor deseado 5. Se observa un nuevo gráfico en la pestaña cada diez segundos
Escenario alternativo	No procede

Tabla 41: CU-08

CU-09	Manipular mapa
Actor	Personal
Objetivo	Visualizar el mapa según las necesidades del usuario
Precondiciones	Acceder al mapa de una zona concreta
Postcondiciones	El personal visualiza el mapa a su gusto
Escenario básico	1. Se selecciona el tipo de zona 2. Se selecciona una zona concreta 3. Se visualiza el mapa de esa zona 4. Se selecciona el tipo de vista que se desea 5. Se selecciona el zoom para el mapa
Escenario alternativo	No procede

Tabla 42: CU-09

CU-10	Autenticarse
Actor	Administrador del sistema
Objetivo	El administrador accede a la parte privada de la aplicación
Precondiciones	El administrador debe tener una cuenta de acceso
Postcondiciones	El administrador entra al contenido privado de la aplicación
Escenario básico	1. Se accede al apartado del servidor (localhost) 2. Se introduce un nombre de usuario y contraseña válidas 3. Si la autenticación es correcta se accede a la parte privada
Escenario alternativo	3b. Si el nombre de usuario o contraseña no son válidos se deberá introducir de nuevo

Tabla 43: CU-10

CU-11	Gestionar aplicación
Actor	Administrador del sistema
Objetivo	El administrador deberá comprobar el buen funcionamiento y que todos los datos mostrados sean correctos
Precondiciones	El administrador deberá autenticarse en el sistema
Postcondiciones	El buen funcionamiento de la aplicación
Escenario básico	1. Se accede al apartado del servidor (localhost) 2. Se introduce un nombre de usuario y contraseña válidas 3. Si la autenticación es correcta se accede a la parte privada 4. Se navega por la página comprobando la información mostrada
Escenario alternativo	No aplica

Tabla 44: CU-11

CU-12	Crear datos en BD
Actor	Administrador del sistema
Objetivo	Insertar nuevos datos en la BD
Precondiciones	El administrador se ha autenticado en el sistema y accede a la base de datos
Postcondiciones	La base de datos ha sido modificada
Escenario básico	1. Se accede al apartado del servidor (localhost) 2. Se introduce un nombre de usuario y contraseña válidas 3. Si la autenticación es correcta se accede a la BD 4. Se introducen nuevas tuplas de datos en las tablas correspondientes
Escenario alternativo	4b. La introducción de nuevos datos no se ha realizado con la sintaxi adecuada en SQL

Tabla 45: CU-12

CU-13	Modificar datos en BD
Actor	Administrador del sistema
Objetivo	Modificar datos que estaban almacenados previamente
Precondiciones	Existen datos almacenados en la BD que se desean modificar
Postcondiciones	Los datos han sido modificados
Escenario básico	1. Se accede al apartado del servidor (localhost) 2. Se introduce un nombre de usuario y contraseña válidas 3. Si la autenticación es correcta se accede a la BD 4. Se accede a la tabla de la que se desea modicar los datos 5. Se accede a la tupla de datos que se desea modificar para editar 6. Se edita al tupla seleccionada
Escenario alternativo	5b. Se selecciona eliminar la tupla de datos y se genera una nueva

Tabla 46: CU-13

CU-14	Eliminar datos en BD
Actor	Administrador del sistema
Objetivo	Se eliminan datos que ya no son válidos de la BD
Precondiciones	Los datos que se desean eliminar se encuentran en la BD
Postcondiciones	Los datos han sido eliminados
Escenario básico	1. Se accede al apartado del servidor (localhost) 2. Se introduce un nombre de usuario y contraseña válidas 3. Si la autenticación es correcta se accede a la BD 4. Se accede a la tabla de la que se desea modicar los datos 5. Se accede a la tupla de datos que se desea eliminar 6. Se marca la opción eliminar y se verifica la opción
Escenario alternativo	5b. Se accede a otra tupla de datos que no se deseaba eliminar 6b. No se verifica la opción eliminar, y por tanto los datos no se eliminan

Tabla 47: CU-14

3.2 Diseño

3.2.1 Base de datos

3.2.1.1 Modelo E/R

El modelo E/R o Entidad-Relación es una herramienta utilizada para el modelado de los datos en un sistema de información. En el diagrama se reflejan las entidades relevantes del sistema así como sus interrelaciones y propiedades. El objetivo del modelo entidad-relación es representar los objetos de la base de datos como entidades con atributos y que se vinculan entre ellas mediante relaciones.

Se considera entidad a cualquier tipo de objeto o concepto sobre el que se recoge información. Las entidades se representan gráficamente mediante rectángulos y su nombre, identificativo y unívoco, aparece en el interior.

Las entidades están formadas por atributos. Un atributo es una característica de interés o un hecho sobre una entidad o sobre una relación. Los atributos representan las propiedades básicas de las entidades y de las relaciones. Gráficamente se representan mediante bolitas que cuelgan de las entidades o relaciones a las que pertenecen. Como breve explicación de los atributos representados en el gráfico podemos decir que los atributos principales se representan con círculo negro y los opcionales con una línea discontinua. Otro tipo de atributos son los multievaluados, es decir, se pueden almacenar diferentes datos en un mismo atributo, estos serán representados con la línea continua y una flecha.

Una relación es la correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. Las relaciones se representan gráficamente mediante rombos y su nombre identificativo aparece en el interior. Un tipo especial de relación que se ha utilizado en el modelo es la herencia, la herencia es un tipo de relación entre entidad "padre" y una entidad "hijo". La entidad "hijo" hereda todos los atributos y relaciones de la entidad "padre". Esta relación se representa mediante un triangulo, la entidad "padre" se encuentra en el vértice superior del triangulo y las entidades "hijo" se conectarán por la base del triangulo.

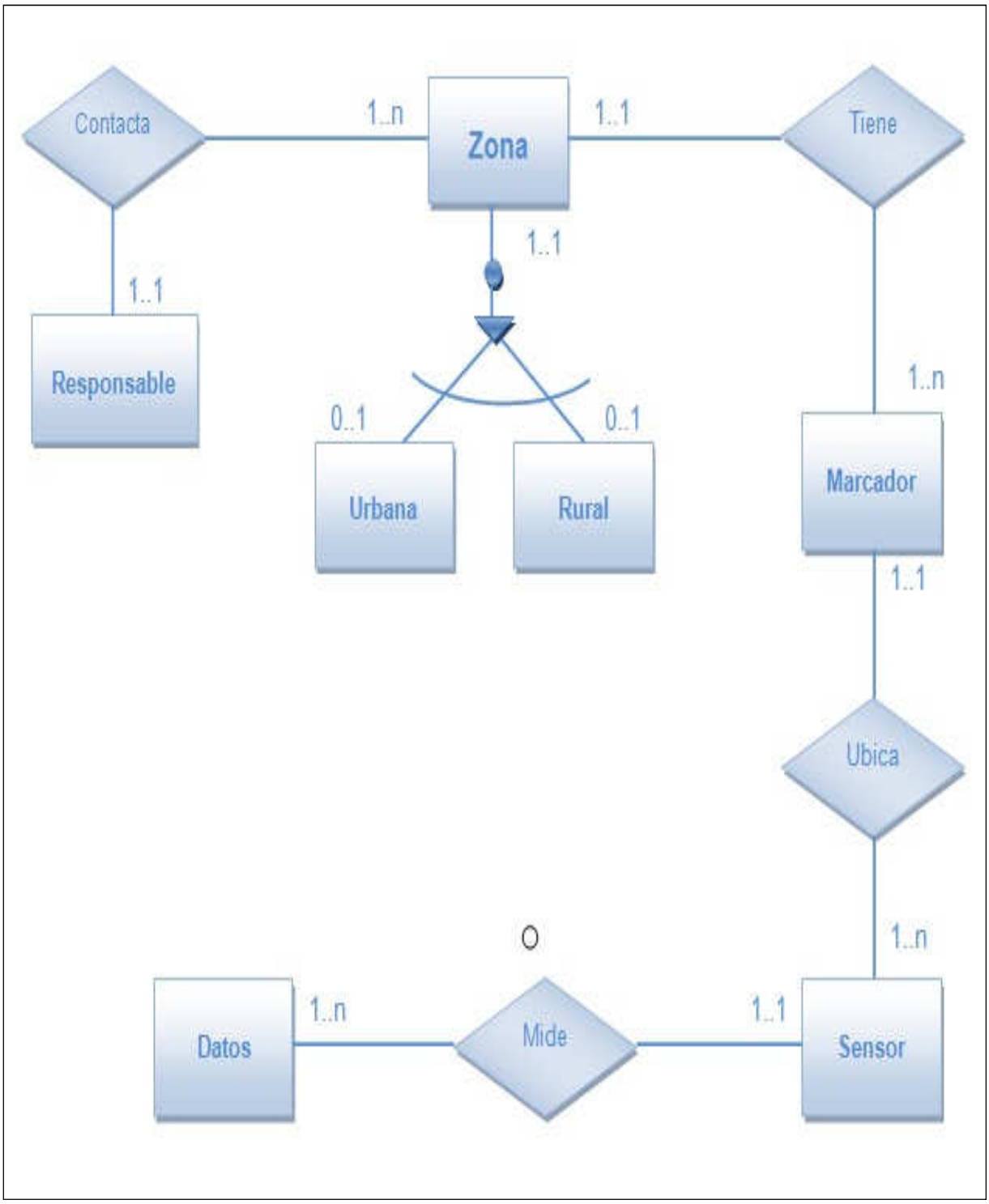


Ilustración 6: Modelo E/R

Las distintas mediciones que se recogen y se reflejan tanto en la base de datos como en el página web se encuentran divididas por zonas, estas zonas pueden ser exclusiva y únicamente rurales o urbanas. Las zonas rurales son zonas de vegetación y para ellas se han contemplado los tipos de cultivo, parque forestal y reserva natural. Todas las zonas tienen campos en común y campos.

Cada una de las zonas está controlada por un responsable, para el cual se almacenan los datos de interés de la persona a contactar en caso de producirse alguna incidencia en alguna de las mismas. Para cada zona se contactará con un único responsable, aunque este pueda ser responsable de más de una zona. En caso de que un responsable sea borrado de la base de datos, no se borrarán los datos de la zona a la que estaba relacionado, sino que el campo Id_Responsable de la tabla Zona quedará a *null* hasta encontrar un nuevo responsable.

Todas las zonas tienen entre 1 y n marcadores, esos marcadores gracias a los campos de latitud y longitud quedarán en un punto exacto del mapa dentro de cada zona representada. Los marcadores representan puntos estratégicos de las zonas, donde se encontrarán los sensores que medirán factores a tener en cuenta para el buen estado de la zona a tratar.

Los sensores se encuentran en los puntos señalados por los marcadores. En un mismo marcador puede haber de 1 a n sensores puesto que cada uno de los sensores mide un dato distinto (temperatura, humedad, luminosidad...).

Los datos se recogen por los sensores de forma continua en un intervalo de tiempo que se programará dependiendo de la necesidad de la zona y del valor a evaluar. Estos datos se almacenan en la tabla datos y son identificados unívocamente por el momento en el que fueron tomados junto con el Id. del sensor al que pertenecen.

3.2.1.2 Modelo Relacional

El modelo relacional para la gestión de una base de datos es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Su idea fundamental es el uso de las relaciones.

El modelo relacional fue propuesto por E.F. Codd en los laboratorios de IBM en California. Se trata de un modelo lógico [Irene Luque Ruiz - Ed. Ra-ma] que establece una estructura sobre los datos, aunque éstos posteriormente puedan ser almacenados de múltiples formas para aprovechar características físicas concretas de la máquina sobre la que se implante la base de datos realmente.

En este modelo todos los datos son almacenados en relaciones, y como cada relación es un conjunto de datos, el orden en que estos se almacenen no tiene relevancia. Esto tiene como ventaja la facilidad para entender y utilizar por un usuario no experto, la información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

A continuación se van a definir las pautas seguidas para la representación del esquema que ayudarán al entendimiento del mismo:

- las relaciones son representadas en negrita, con un identificador único y entre paréntesis se sitúan los atributos que contendrá cada tupla de esa relación.

- Los atributos de cada relación están separados por comas, pueden ser opcionales y no tomar ningún valor, en tal caso se indicará con un asterisco.

- Aquellos atributos que sean clave primaria de la tabla aparecerán subrayados, y los que representen clave ajena aparecerán con una flecha inferior que estará apuntando a la tabla a la que ese atributo está referenciando. A cada flecha le estará asignado un texto que definirá las reglas de borrado y modificación de las claves ajenas. Por un lado la regla de borrado elegida es (DC-*delete cascade*) que sin duda es la menos restrictiva, excepto en uno de los casos en el que hemos elegido (DSN-*delete set null*) donde la tabla a borrar no será eliminada, los valores no válidos serán sustituidos por valores nulos. Por otro lado, la regla de modificación se realizará en todos los casos en cascada (UC-*update cascade*).

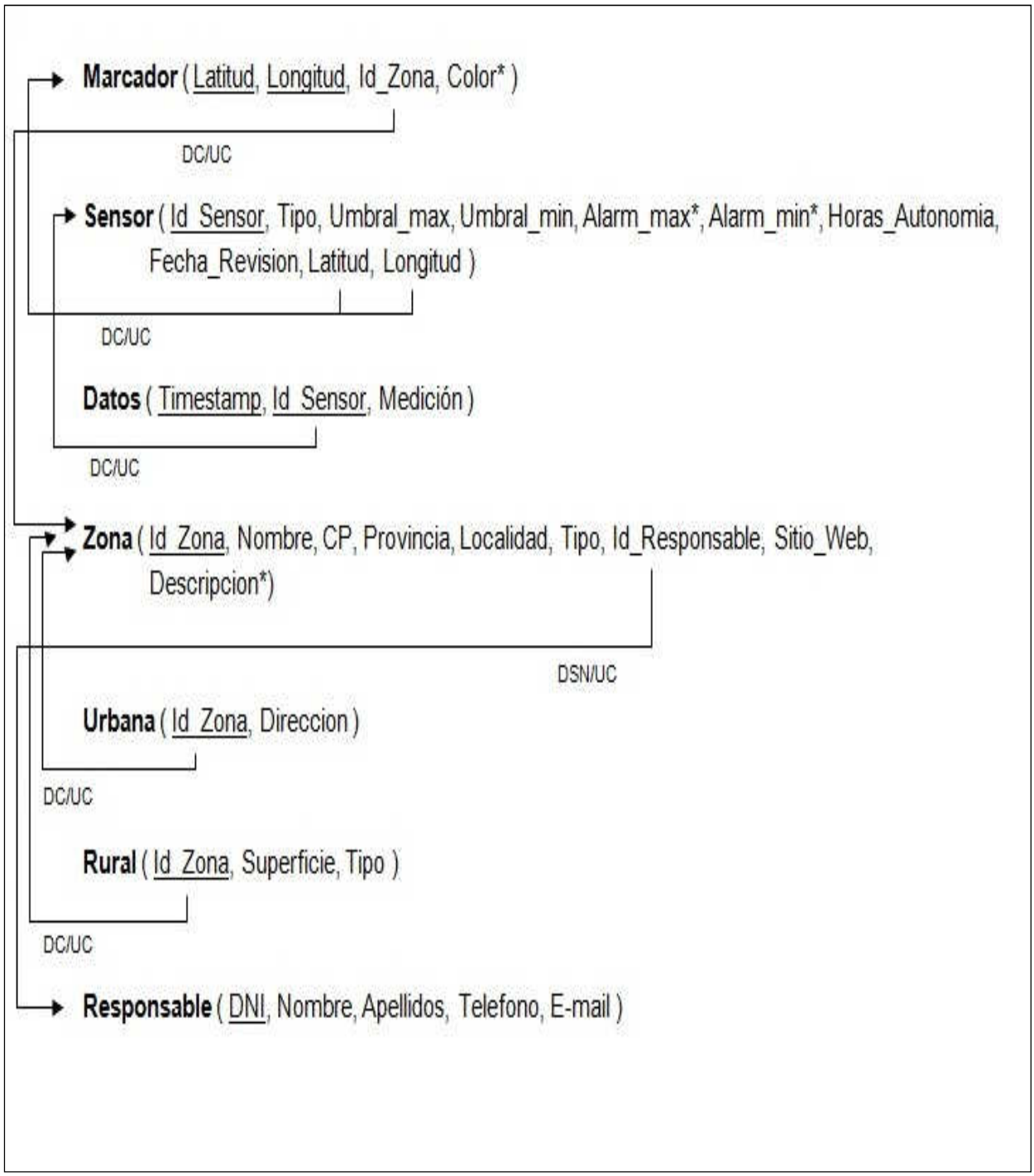


Ilustración 7: Esquema relacional

- **MARCADOR**: Esta tabla contiene todos los datos pertenecientes a los puntos geográficos donde se encuentran los sensores posicionados en el mapa. Se requiere de la unión de la Latitud y la Longitud para diferenciar los posibles marcadores, por este motivo tiene dos claves primarias, dos marcadores pueden tener la misma latitud o la misma longitud, pero no las dos características unidas. Como clave ajena, ID_Zona sirve para identificar los marcadores almacenados en los diferentes mapas, y por tanto zonas posibles y de este modo facilitar las búsquedas. En esta tabla se almacenan todas las posiciones de todos los sensores en las distintas zonas:

    - ***Latitud***: Indica la latitud en la que está situado el marcador. Es de tipo decimal, con la restricción de que solo puede tener 6 decimales, de esta forma disminuimos el tamaño de la variable sin restringir las posibles latitudes, puesto que ese número de decimales es suficiente para posicionar sin error. No puede tomar valores nulos por ser clave primaria.

    - ***Longitud***: Indica la longitud en la que está situado el marcador. Es de tipo decimal, con la restricción de que solo puede tener 6 decimales, de esta forma disminuimos el tamaño de la variable sin restringir las posibles longitudes, puesto que ese número de decimales es suficiente para posicionar sin error. No puede tomar valores nulos por ser clave primaria.

    - ***Id_Zona***: Clave ajena que relaciona a cada marcador con la zona en la que se encuentra. Es de tipo *Varchar* con un máximo de tres dígitos, lo cual se considera suficiente para identificar todas las zonas existentes. No puede tomar valores nulos por ser clave ajena con borrado y modificación en cascada.

    - ***Color***: Campo para almacenar el color que toma el marcador en el mapa. El campo es un Varchar con un máximo de 20 dígitos. Este campo es opcional, puede tomar valores nulos.

- **SENSOR:** La tabla sensor almacena toda la información referente al mecanismo utilizado para realizar mediciones. Como clave primaria se encuentra el Id_Sensor, con un valor identificativo para cada uno de los mismos. Como claves ajenas se encuentran la Latitud y la Longitud, ambas claves primarias de la tabla marcador. Sensor está relacionada con marcador puesto que en un marcador se pueden encontrar de uno a n sensores y cuando se necesitan los datos de un sensor, se necesita posicionar.

    - ***Id_Sensor***: Clave primaria de la tabla sensor, es un campo de identificación. Está formado por un Varchar de longitud máxima 3. . No puede tomar valores nulos por ser clave primaria.

    - ***Tipo***: Atributo para conocer el tipo de medición que realiza el sensor, puede ser de humedad, temperatura, luminosidad o sonido. Está formado por un Varchar de 20 dígitos como máximo. No puede tomar valores nulos.

    - ***Umbral_min***: Dato mínimo que el sensor puede tomar, de modo que si tomara valores aún menores no serían medidos. El atributo es un Smallint de seis dígitos. No puede tomar valores nulos.

    - ***Umbral_max***: Dato máximo que el sensor puede tomar, de modo que si tomara valores aún mayores no serían medidos. El atributo es un Smallint de seis dígitos. No puede tomar valores nulos.

    - ***Alam_min***: Este atributo sirve para conocer la temperatura mínima a partir de la cual se considera que los datos tomados están fuera de la normalidad. Está definido como un *Smallint* de seis dígitos. Es un campo opcional, puede tomar valores nulos.

    - ***Alarm_max***: Este atributo sirve para conocer la temperatura máxima a partir de la cual se considera que los datos tomados están fuera de la normalidad. Está definido como un smallint de seis dígitos. Es un campo opcional,.

- ***Horas_Autonomia***: Número de horas que el sensor estará funcionando, dependiendo de las características y la batería del modelo utilizado. Está definido como un smallint de seis dígitos. No puede tomar valores nulos.
- ***Fecha_Revision***: Fecha en la que se realizó la última supervisión al sensor para comprobar su correcto funcionamiento. Es del tipo date. No puede tomar valores nulos.
- ***Latitud***: Clave ajena que referencia a la clave primaria de la tabla marcador, Latitud. Es de tipo decimal, igual que en la tabla referenciada. Solo puede tomar valores de latitudes creadas en la tabla marcador, no puede tomar valores nulos.
- ***Longitud***: Clave ajena que referencia a la clave primaria de la tabla marcador, Longitud. Es de tipo decimal, igual que en la tabla referenciada. Solo puede tomar valores de longitudes creadas en la tabla marcador y que correspondan con la latitudes tanto de la tabla marcador como de sensor. No puede tomar valores nulos.

- **DATOS:** Esta tabla almacena todos los valores tomados por todos los sensores que se encuentran en cada una de las zonas representadas, sin ninguna restricción. Esta tabla posee como clave primaria la unión de dos de sus atributos que son Timestamp e Id_Sensor, puesto que se pueden recoger dos tuplas de datos en el mismo instante pero no en el mismo instante y por el mismo sensor. Además, Id_Sensor será la clave ajena que referenciará a la tabla Sensor para identificar aquel que ha recogido los datos.

    - ***Timestamp***: Almacena el momento exacto en el que se ha recogido una medición. Es del tipo timestamp y es la clave primaria de la tabla. No puede tomar valores nulos por ser clave primaria.

    - ***Id_Sensor***: Almacena el identificador del sensor que ha recogido el dato, es del tipo varchar con tres dígitos como máximo y además de ser clave primaria es clave ajena puesto que referencia a la tabla Sensor del que se haya recogido la medición. No puede tomar valores nulos por ser clave primara.

    - ***Medición***: Es el valor tomado por el sensor. Es del tipo float y en ningún caso puede ser nulo, puesto que si no se ha recogido ningún valor no tiene sentido almacenar los datos.

- **ZONA:** Esta tabla recoge los datos referidos a las zonas de medición, aquellas representadas en la aplicación. Como clave primaria tiene un campo de identificación, Id_Zona, y como clave ajena tiene el campo Id_Responsable para referenciar a la tabla de Responsable.

  - ***Id_Zona***: Es la clave primaria de la tabla. Como todos los identificadores en un Varchar de longitud tres dígitos. No puede tomar valores nulos por ser clave primaria.

  - ***Nombre***: Es el nombre de la zona tratada. Es un Varchar de cincuenta dígitos de longitud. No puede tener valor nulo.

  - ***CP***: Es el código postal de la zona. Es del tipo Smallint de cinco dígitos pero sin signo. No puede tener valor nulo.

  - ***Provincia***: Almacena la provincia en la que se encuentra la zona tratada. Es del tipo Varchar con una longitud de veinticinco dígitos. No puede tener valor nulo.

  - ***Localidad***: Almacena la localidad en la que se encuentra la zona tratada. Es del tipo Varchar con una longitud de veinticinco dígitos. No puede tener valor nulo.

  - ***Tipo***: Indica el tipo de zona a la que pertenece. Este atributo es un enumerado y solo puede tomar los valores de "rural" o "urbana".

  - ***Id_Responsable***: Clave ajena que relaciona las tablas Zona y Responsable. Es del tipo Varchar de nueve dígitos. Puede tomar valores nulos puesto que es posible que a una zona no se le haya asignado responsable.

  - ***Direccion_Web***: Dirección web ajena a la aplicación, donde se pueden encontrar datos sobre las zonas. Es del tipo Varchar con una longitud máxima de treinta.

- ***Descripción***: Definición de la zona. Es del tipo text y puede tener valor nulo, es un campo opcional.

- **URBANA:** Esta tabla es la tabla "hijo" de la tabla Zona, por tanto está relacionada con Zona por el atributo de identificación, ya que una zona urbana tendrá todos los atributos de zona más los propios de urbana.

  - ***Id_Zona***: Campo de identificación para relacionar la tabla urbana con zona. Es del tipo Varchar con longitud máxima de tres dígitos. No puede tomar valores nulos al ser clave primaria.

  - ***Dirección***: Al ser una zona urbana dispone de una dirección para su fácil localización. Este atributo es del tipo Varchar de máximo cuarenta caracteres. Su valor no puede ser nulo.

- **RURAL:** Es la tabla "hijo" de la tabla Zona, por tanto tendrá atributos propios pero todos los registros de esta tabla tendrán atributos heredados de la tabla padre. Tiene un atributo de identificación como clave primaria, para una esta tabla con la tabla Zona.

  - ***Id_Zona***: Campo de identificación para relacionar la tabla rural con zona. Es del tipo Varchar con longitud máxima de tres dígitos. No puede tener valores nulos por ser clave primaria.

  - ***Superficie_m2***: Este atributo almacena la superficie de la zona tratada en metros cuadrados. Es del tipo Mediumint, con una longitud máxima de ocho dígitos y sin signo. Su valor no puede ser nulo.

  - ***Tipo***: Este atributo identifica el tipo de zona rural dentro de unas opciones. Es del tipo enumerado, y sus valores solo pueden ser; Cultivo, Parque forestal o Reserva natural. Su valor no puede ser nulo.

- **RESPONSABLE:** Esta tabla almacena los datos referentes al personal responsable de cada una de las zonas. Cada una de las zonas está vigilada por un responsable, aunque en el caso de eliminar a alguno, no se eliminaría la zona, sino que se marcaría a nulo el campo que las relaciona y el administrador del sistema le asignará un nuevo responsable a la zona.

  - ***DNI***: Campo de identificación de cada uno de los responsables. Es del tipo Varchar de nueve. Su valor no puede ser nulo por ser clave primaria.

  - ***Nombre***: Nombre del responsable. Es del tipo Varchar con una longitud máxima de veinte. Su valor no puede ser nulo.

  - ***Apellidos***: Apellidos del responsable. Es del tipo Varchar con una longitud máxima de cuarenta. Su valor no puede ser nulo.

  - ***Teléfono***: Teléfono de contacto del responsable. Es del tipo *Integer* con una longitud máxima de nueve. Su valor no puede ser nulo.

  - ***Email***: Dirección web del responsable. Es del tipo Varchar de longitud máxima treinta. Su valor no puede ser nulo.

3.2.2 Aplicación

3.2.2.1 Diagrama de navegación

Diseñar el diagrama de navegación consiste en describir dinámicamente el orden y tipo de pantallas que se van produciendo durante la ejecución de la aplicación. Consiste en un autómata finito que describe la transición de ventanas para conocer de este modo las ventanas que preceden y a las que se pueden acceder desde cada una de las mismas.

En la siguiente figura se puede observar el diagrama de navegación de la aplicación realizada, de este modo se puede conocer con facilidad el número de ventanas de las que consta nuestra aplicación así como los pasos necesarios para conseguir el fin deseado.

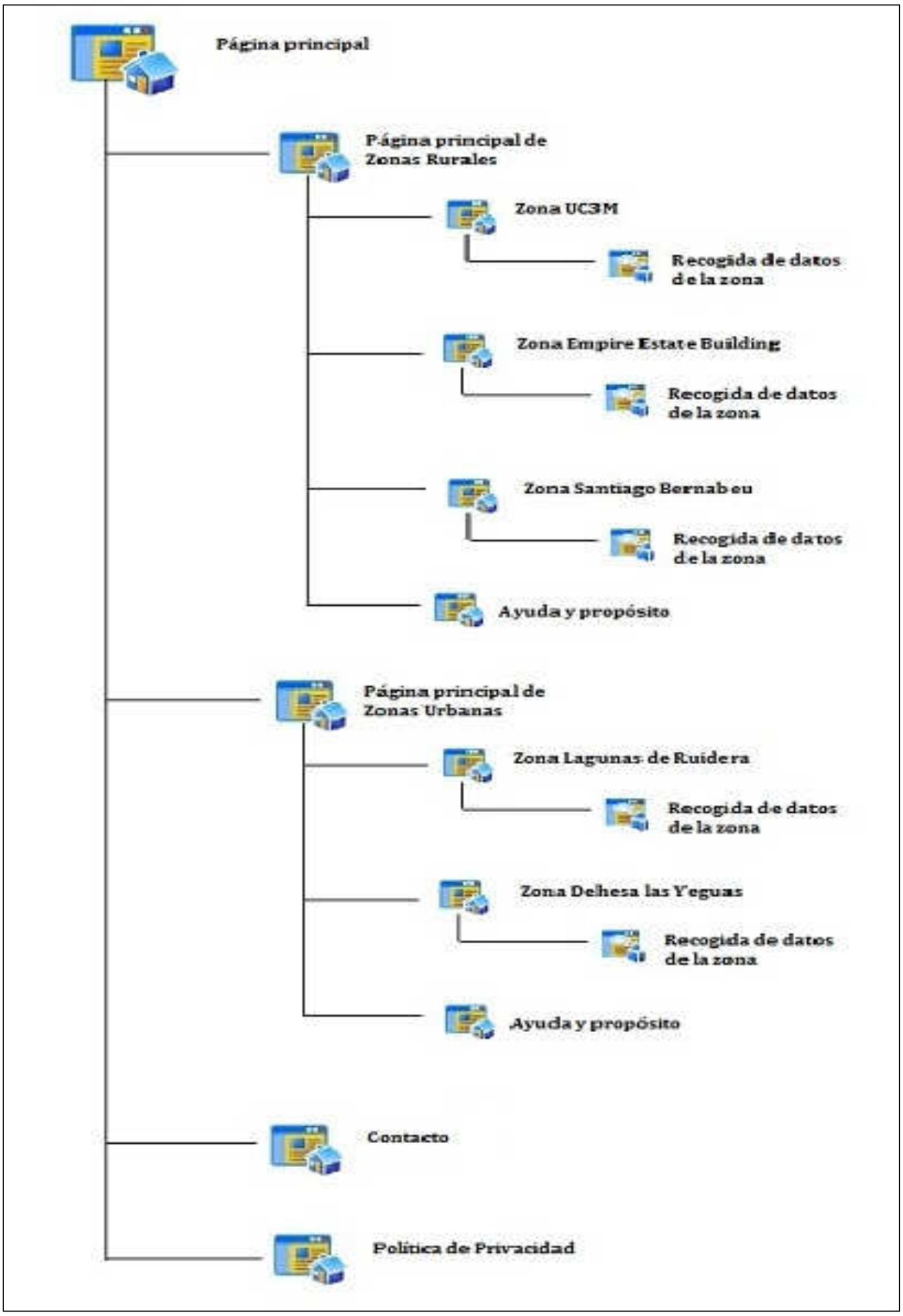


Ilustración 8: Diagrama de navegación

En el diagrama anterior se puede analizar la aplicación realizada. Hay una pantalla principal en la que además de los enlaces hacia la información de los Contactos y la Política de privacidad que son únicamente informativos, se ha de elegir entre la principal división de la que consta el sitio web. Zona Rural o Urbana.

Si se elige una Zona Rural se accede a una pantalla intermedia donde se muestran todas las posibles zonas rurales a visualizar. Cada una de las cuales lleva a otra página en la que muestra la información relacionada con la zona seleccionada, se mostrará una breve descripción de la zona y un mapa de cada una. Lo mismo ocurre al elegir una zona Urbana.

3.3 Implementación

3.3.1 Base de datos

Para tratar y explicar la implementación de la base de datos propuesta hay que referirse al lenguaje SQL. éste ha sido el nexo entre el diseño lógico realizado y el sistema gestor de bases de datos.

Para definir la estructura de la base de datos se ha utilizado el lenguaje LDD y para el manejo de la estructura el LMD. En este punto nos vamos a centrar en los pasos seguidos para la implantación y las principales sentencias utilizadas para especificar la estructura de los datos, por tanto nos centraremos en sentencias del lenguaje LDD.

3.3.1.1 Crear objetos

Partiendo del diseño lógico, el primer paso es crear las tablas en concordancia. Para ello se ha utilizado la sentencia "CREATE TABLE".

```
CREATE TABLE 'TABLA_NOMBRE' (
'CAMPO_1' tipo,
'CAMPO_2' tipo )
```

Esta sentencia requiere principalmente de tres atributos; un nombre, una definición y unas opciones. Hay numerosas opciones de tabla pero en nuestro caso hemos utilizado únicamente la sentencia "Type=InnoDB" para modificar el motor de almacenamiento que usará la tabla y poder así, utilizar claves ajenas.

La definición de la estructura de la tabla se compone de dos partes bien diferenciadas; la declaración de las columnas y las restricciones de claves o índices.

La declaración de columnas consta de tres apartados: el nombre de la columna, el tipo y la posibilidad de indicar si son posibles los valores nulos.

Todas las tablas creadas y los atributos de cada una de ellas han sido definidos y explicados en apartados anteriores. En este punto nos vamos a centrar en definir todos los tipos de datos que han sido necesarios para la implementación:

- Decimal: Número en coma flotante, el número se almacena como una cadena y es posible especificar el número de decimales que guardar.
- SmallInt: Número entero con o sin signo. Con signo el rango de valores va desde -32768 a 32767. Sin signo, el rango de valores es de 0 a 65535. Su tamaño de almacenamiento es de 2bytes.
- Integer: Número entero con o sin signo. Con signo el rango de valores va desde -2147483648 a 2147483647. Sin signo el rango va desde 0 a 429.4967.295. Su tamaño de almacenamiento es de 4bytes.
- MediumInt: Número entero con o sin signo. Con signo el rango de valores va desde -8.388.608 a 8.388.607. Sin signo el rango va desde 0 a16777215. Su tamaño de almacenamiento es de 3bytes.
- Varchar: Almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres. Su tamaño de almacenamiento es de n + 1 bytes.
- Enum: Campo que puede tener un único valor de una lista que se especifica. El tipo Enumerado acepta hasta 65535 valores distintos. Su tamaño de almacenamiento puede ser de 1 o 2 bytes dependiendo del número de valores.
- Text: Un texto con un máximo de 65535 caracteres. Su tamaño de almacenamiento es de la longitud mas 2 bytes.

- Timestamp: Es la combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 2037. El formato de almacenamiento depende del tamaño del campo. En nuestro caso, al tener el formato AñoMesDia aammdd el tamaño es de 6 bytes.
- Date: Tipo fecha, almacena una fecha. El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. El formato de almacenamiento es de año-mes-día y su tamaño de 3 bytes.

En el anexo se puede consultar la sintaxis utilizada para la creación de las tablas en la base de datos que nos ocupa.

3.3.1.2 Consultas

Las consultas son necesarias para obtener la salida de datos en la aplicación Web. Las consultas implementadas para obtener la información de la base de datos son bastante simples y no utilizan ningún operador de conjuntos, debido a que el diseño ha sido realizado para facilitar la implementación.

Para realizar las consultas se ha utilizado el LMD del lenguaje SQL, concretamente el operador 'SELECT', en el cual, indicando el nombre de la columna o columnas que se quieren seleccionar y la tabla de origen de los datos nos permite obtener lo deseado de una manera sencilla.

***SELECT** [**DISTINCT**] nombre_columna/s*
***FROM** nombre_tabla*
*[**WHERE** {condicion}]*
*[**ORDER BY** nombre_columna [**DESC**]]*

Esta sentencia permite una serie de opciones añadidas a lo anterior. Se pueden indicar premisas para que únicamente se seleccionen registros que cumplan una determinada condición, así como ordenar el resultado de salida, ya sea ascendentemente o descendentemente (añadiendo la palabra 'DESC').

Las consultas se han integrado en el código PHP desarrollado para la aplicación web. En el formato de presentación de las consultas aparecen palabras con el símbolo '$? delante, estas palabras representan variables ya que muchas de las consultas varían en función de un determinado parámetro que generalmente se corresponde con un valor obtenido de una consulta anterior.

3.3.2 Aplicación web

La aplicación web se ha implementado principalmente utilizando el lenguaje HTML y la hoja de estilos CSS. Para aumentar las funcionalidades se han utilizado tecnologías algo más complejas que ya has sido comentadas en apartados anteriores, a continuación vamos a describir los métodos llevados a cabo para implementar los aspectos más relevantes.

3.3.2.1 Insertar los mapas de Google en la web

En este apartado se explica paso a paso como se han insertado los mapas de Google Maps en nuestra aplicación web utilizando las Api de Google.

El primer paso es obtener una clave de Google Maps Api key, que es gratuita y te permite disfrutar de los servicios de Google Maps. Para conseguirla hay que acceder a la dirección http://www.google.com/apis/maps y seguir las instrucciones de "Sign up for a google Api key". Será necesario tener una cuenta de Gmail y poner el nombre del dominio donde se pretende usar el mapa, de esta manera Google da los permisos necesarios para el uso de los mapas.

Una vez conseguida la clave, Google proporciona un código que se debe de insertar en el archivo *.html donde se encuentre la pagina en la que se desea mostrar el mapa, dentro de la cabecera <head>, con una etiqueta <script>. Para añadir el mapa en la página hay que enlazar con el fichero javascript que se necesita para ejecutar el mapa y decirle a Google que la clave introducida es correcta.

```
<script src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=false&amp;
key=ABQIAAAA4il-n9MJdijb5zDIqrXr5hT2yXp_ZAY8_ufC3CFXhHIE1NvwkxQ_moyRhO3oKw1aC9kfkrCmViOXpw"
type="text/javascript"></script>
```

Ilustración 9: Insertar la clave proporcionada por Google Maps

A continuación se inserta un *div* para que el mapa aparezca en pantalla. Estará en la parte de la página donde se quiera que aparezca el mapa y hará referencia a la hoja de estilo CSS para seleccionar las características del mapa a mostrar:

```
<div id="mapa_principal">

    </div>

#mapa_principal {
border: 1px solid black;
width: auto;
height: 400px;
margin: 10px 10px 10px 10px;
```

Ilustración 10: Insertar el mapa en la página

Lo siguiente es introducir el código de Google para cargar el mapa en la aplicación y posicionarlo en la zona deseada, este código será introducido en un fichero a parte del tipo *.js:

```
var mapa;

function initialize() {
    //google.load("visualization", "1", {packages:["piechart"]});
    //google.setOnLoadCallback(drawChart);


    if (GBrowserIsCompatible()){
        mapa = new GMap2(document.getElementById("mapa_principal"));

        //Controles añadidos en el mapa y lugar donde aparece centrado.
        mapa.setCenter(new GLatLng(40.332397, -3.765671), 15);
        mapa.addControl(new GSmallZoomControl());
        mapa.addControl(new GMapTypeControl());
        mapa.addControl(new GOverviewMapControl());
```

Ilustración 11: Inicializar el mapa

En el caso que se deseen añadir marcadores al mapa habrá que inicializarlos y posicionarlos en un punto deseado del mapa, a esos marcadores se le pueden añadir numerosas funcionalidades de movimientos y un gran número de imágenes para identificarlos.

```
//Datos del marcador 1
var pos_sensor1 = new GLatLng(40.332538, -3.767336);
var opt = {title:"Sensor 1", draggable:false};
var sensor1 = new GMarker(pos_sensor1,opt);

//Añadimos los marcadores al mapa
mapa.addOverlay(sensor1);
```

Ilustración 12: Insertar marcadores en el mapa

Una vez definidos los marcadores, es posible asignarle datos a cada uno de ellos mediante el uso de ventanas asignadas a los diferentes marcadores. Cada una de las ventanas puede tener diferentes pestañas para diferentes datos asignados, pero como limitación todo el texto introducido en cada una de las pestañas debe ser HTML.

```
//Definicion de variables que mostraremos en las pestañas de cada marcador
var tab10 = new GInfoWindowTab("Sensor 51", "<iframe src='s51.html' width='460px' height='260px' scrolling='no' frameborder='0'>
var tab11 = new GInfoWindowTab("Sensor 52", "<iframe src='s52.html' width='460px' height='260px' scrolling='no' frameborder='0'>
var tab20 = new GInfoWindowTab("Sensor 53", "<iframe src='s53.html' width='460px' height='260px' scrolling='no' frameborder='0'>
var tab21 = new GInfoWindowTab("Sensor 54", "<iframe src='s54.html' width='460px' height='260px' scrolling='no' frameborder='0'>
var tab30 = new GInfoWindowTab("Sensor 55", "<iframe src='s55.html' width='460px' height='260px' scrolling='no' frameborder='0'>


var arrayTab1 = [tab10, tab11];
var arrayTab2 = [tab20, tab21];
var arrayTab3 = [tab30];

//Para "bindear" un array de pestañas a cada marcador
sensor1.bindInfoWindowTabsHtml(arrayTab1);
sensor2.bindInfoWindowTabsHtml(arrayTab2);
sensor3.bindInfoWindowTabsHtml(arrayTab3);
```

Ilustración 13: Definir ventanas para cada marcador

De este modo, obtenemos los mapas de Google Maps para mostrar en nuestra aplicación web, en la parte de la página que se desee. Se tendrá un mapa por cada una de las zonas representadas, y en cada uno de los mapas se encontrarán representados los marcadores. Cada uno de estos marcadores simboliza un punto estratégico de la zona tratada donde se encuentran un número determinado de sensores posicionados.

Ilustración 14: Mapa de la zona Uc3m

Como aspecto a tener en cuenta en relación a los marcadores posicionados en el mapa cabe destacar la falta de precisión. Los marcadores se sitúan según una latitud y una longitud, estos datos son suficientes si se desean posicionar los marcadores en zonas rurales que se suponen extensiones a lo largo pero nunca a lo alto. El inconveniente aparece a la hora de posicionar los marcadores en un edificio, no existe el parámetro de la altura para especificar, por tanto, ese marcador será orientativo pero nunca real.

Esto se puede comprobar al posicionar los marcadores en el edificio Empire State de Nueva York. El objetivo era situar dos marcadores en la base del edificio y uno en la altura, jugando con las latitudes y longitudes parece que el resultado obtenido era el deseado.

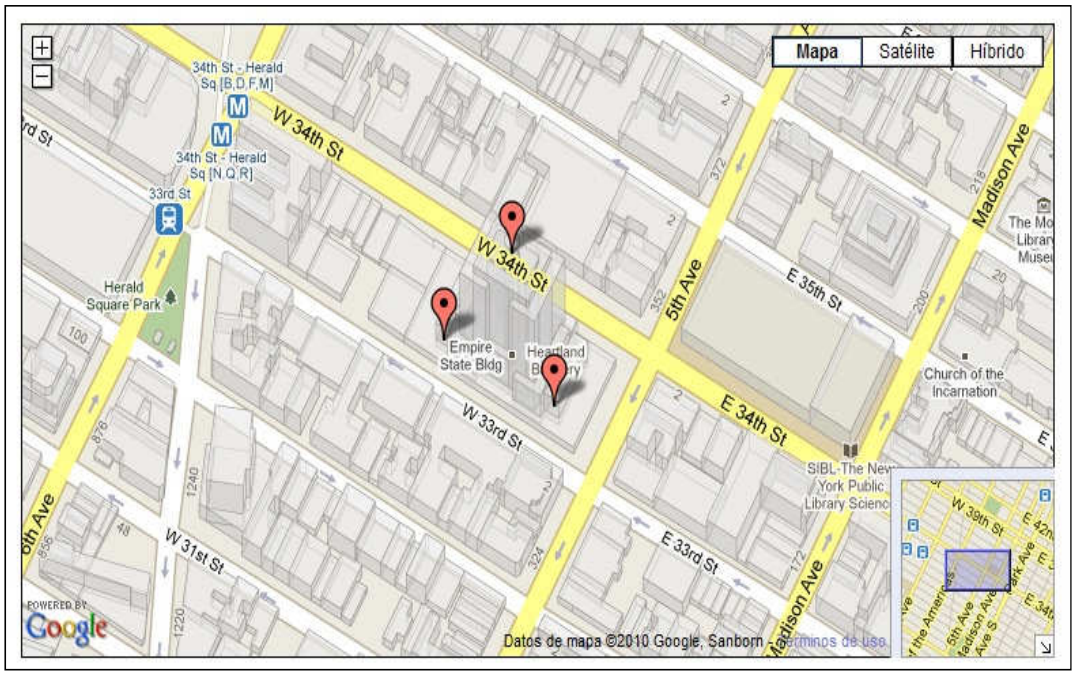


Ilustración 15: Vista del Empire State

En cambio, si se modifica la vista a Satélite, se puede comprobar que el marcador que parece estar en la cima del edificio, en realidad se encuentra en la acera de enfrente. A pesar de estar en una zona del mundo donde se encuentran las últimas actualizaciones en mapas, como es Nueva York, y podemos ver los edificios en 3D, no se puede situar el marcador según una altura, por tanto la posición del marcador que se supone un punto estratégico de posicionar sensores no puede ser del todo correcta pues todavía no es posible trabajar con un parámetro de altura en los mapas.

Ilustración 16: Vista satélite del Empire State

Cada marcador tiene una ventana asociada con un número de pestañas que corresponde al número de sensores que se encuentran en ese mismo punto. Cuando desde la aplicación se pincha en uno de los globos rojos del mapa se abre la ventana de ese globo. En el interior de las ventanas solo se permite introducir código HTML por lo que se muestra una única imagen por pestaña correspondiente a la gráfica representativa de los valores recogidos por cada sensor.

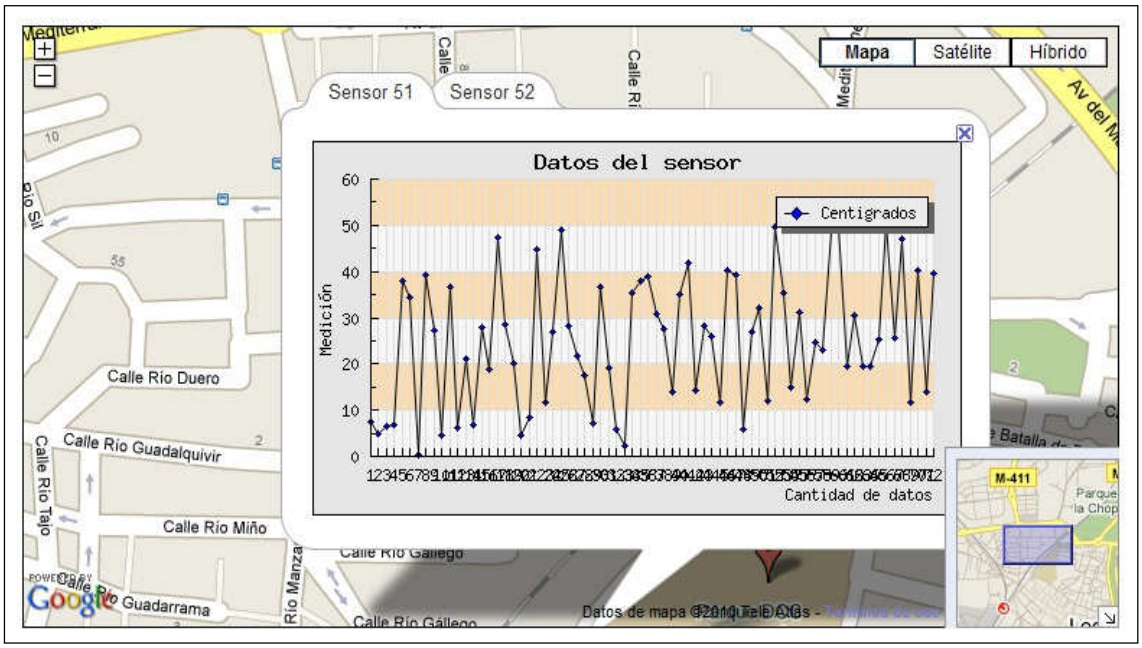


Ilustración 17: Gráfico de datos de cada sensor

3.3.2.2 Insertar datos automáticamente en la base de datos

A partir de la página principal de cada zona se pueden abrir pestañas nuevas en el navegador con una página a la que hemos denominado *emulacion.html* y que puede ser abierta en todas las pestañas que se desee.

Esta página es la que se ha utilizado para actuar de intermediario entre la base de datos, que tiene los valores almacenados, y la aplicación que muestra los valores de forma gráfica.

Ilustración 18: Página de emulación

En esta página se selecciona el número del sensor del que se desea recoger datos y almacenarlos y cuando pinchamos en el botón "Iniciar" comenzará la recogida de los datos.

En principio, este proceso debería estar en permanente funcionamiento, pero tenemos que tener en cuenta que nuestra aplicación no está subida a un servidor, que no contamos con los sensores reales que generen datos y por tanto, no tenemos el software de unión entre el sensor y la base de datos, por lo que se ha tenido que crear este proceso intermedio que emulará el funcionamiento real en caso de tener los sensores y el software que permita relacionarlos con la base de datos.

La ventana de *emulacion.html* es posible abrirla en diferentes pestañas del navegador y en cada una de ellas, seleccionar un número de sensor diferente, del que se desea recoger los datos. Cada vez que damos a "Iniciar" se crea un proceso que actúa por detrás de la aplicación, sin que el usuario pueda percibirlo, almacenando los datos. Esto es posible gracias a la tecnología Ajax.

Cada vez que se inicia un proceso nuevo de recogida de datos se llama a la función insertAjax() que se encuentra en un archivo a parte denominado Ajax.js del tipo JavaScript. Esta función captura el valor del sensor seleccionado y del cual se desea recoger los datos y crea el objeto XMLHttpRequest, con este objeto se realiza el método open, para asignar la dirección url de donde va a cargar los datos, y el método send, para efectuar la llamada a la url asignada con open.

La url asignada para cargar los datos es la de otro fichero distinto, al que hemos denominado logica.php. En este archivo se utiliza la tecnología PHP para conectarnos a la base de datos y almacenar los valores, que en este caso serán aleatorios puesto que no tenemos los sensores físicos.

3.3.2.3 Crear las gráficas de datos

Cada vez que se emula el funcionamiento de un sensor se están generando y almacenando los datos de ese sensor, esos valores almacenados son los que nos permiten crear las gráficas representadas en las pestañas de los mapas gracias a la tecnología JPGraph.

Éstas gráficas son representadas en las pestañas del interior del mapa de cada zona y como se ha comentado en capítulos anteriores, solo admiten código HTML por lo que se nos han restringido las posibles funcionalidades y la imposibilidad de generar otro tipo de gráficos diferentes a los generados por las librerías JPGraph, como podrían ser los proporcionados por Google Chart, pues para ellos se necesitaría utilizar otro tipo de tecnologías como JavaScript.

La implementación realizada para esta funcionalidad consiste en llamar en cada una de las pestañas a un archivo *.html, existe uno de ellos por cada sensor representado en cada uno de los mapas de las diferentes zonas.

El archivo *.html es el encargado de cargar la imagen que utiliza la librería JPGraph para mostrar el gráfico según los datos extraídos de la BD, la carga se realiza en este archivo cada diez segundos y es mostrado en la web principal mediante un iFrame para que las recargas continuadas no entorpezcan la usabilidad de la aplicación. De esta manera, cuando visualizamos el gráfico en la página principal solo notamos un leve parpadeo en el área que ocupa el gráfico.

El gráfico se genera en el archivo *grafico.php*, al llamar a este archivo se devuelve una imagen, dicha imagen es el gráfico ya creado según las características elegidas. En este archivo lo primero que se debe tener en cuenta es referenciar las librerías de JPGraph necesarias y acceder a la BD para obtener los datos de los que se desean obtener el gráfico.

Con las librerías JPGraph es posible realizar gráficos a partir de un array de datos, no desde una consulta, por tanto, los datos obtenidos de la consulta se deben almacenar en una variable de tipo array para poder trabajar con ellos.

Este archivo es único para todos los sensores, pues en el archivo html se almacena el número del sensor del que se quieren obtener los datos y a partir de ese se llama a grafico.php que recibe el número del sensor del que se hace la consulta y se realiza el gráfico. Cada vez que se desea actualizar el gráfico se realiza una nueva consulta a la BD y se genera un gráfico nuevo actualizado.

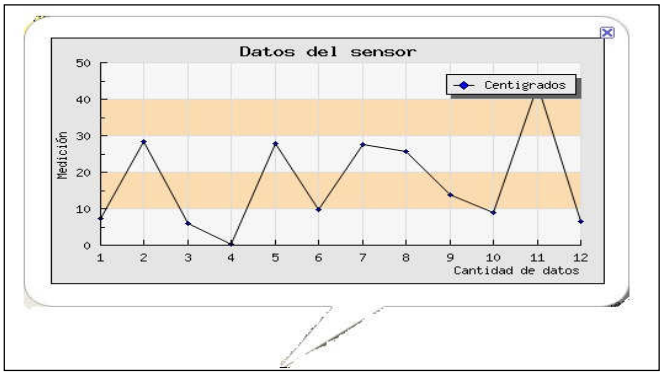


Ilustración 19: Gráfico en la primera actualización

Diez segundos después la ventana se actualiza:

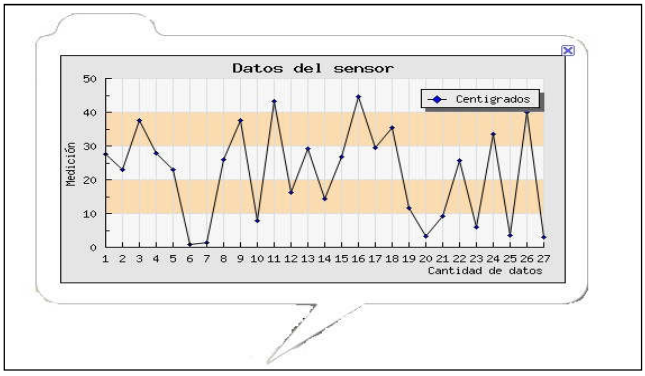


Ilustración 20: Gráfico en la segunda actualización

Capítulo 4

Planificación y Presupuesto

En este capítulo se mostrarán los datos obtenidos para la planificación y presupuesto del proyecto. La planificación se ha llevado a cabo mediante las herramientas de Microsoft Project, en cambio, para el presupuesto se han utilizado las plantillas proporcionadas por la UC3M.

4.1 Planificación

4.1.1 Modelo de gestión del proyecto

El ciclo de vida ha marcado la metodología de trabajo, de principio a fin, del proyecto y la planificación del mismo. El modelo de desarrollo utilizado para la realización de la aplicación ha sido el modelo en espiral (Piattini M. et al, 2003).

El ciclo de vida del desarrollo software es una descripción de las distintas formas de desarrollo de un proyecto, es decir, la orientación que debe seguirse para obtener, a partir de los requisitos del cliente, sistemas que puedan ser utilizados por éste. Este término también se conoce como conjunto de fases, procesos y actividades requeridas para ofertar, desarrollar, probar, integrar, explotar y mantener un producto software.

El ciclo de vida utilizado para el desarrollo define el orden para las tareas o actividades involucradas, también definen la coordinación entre ellas, enlace y realimentación entre las etapas.

El modelo en espiral fue propuesto por Boehm en 1988. Básicamente consiste en una serie de ciclos que se repiten en forma de espiral, comenzando desde el centro. En cada una de las iteraciones hay que tener en cuenta las necesidades que debe cubrir el producto así como las diferentes maneras de conseguir los objetivos de forma exitosa. Si el resultado no es el adecuado o se necesita mejorar se planificarán los siguientes pasos y se comienza un nuevo ciclo de la espiral.

Se ha comentado que el ciclo de vida define el orden para las tareas, por tanto vamos a definir las tareas del ciclo de vida en espiral, para cada ciclo habrá cuatro tareas:

1. Fijar los objetivos, fijar las restricciones e identificar los riesgos del proyecto y las estrategias alternativas para evitarlos.

2. Analizar los riesgos posibles que pueden aparecer en el desarrollo.

3. Desarrollar, verificar y validar las tareas.

4. Planificar el desarrollo. Se revisa todo lo hecho y se decide el modo de continuar y planificar la siguiente actividad.

4.1.2 Diagrama de Gantt

Una vez explicados los aspectos generales de la gestión del proyecto, procedemos a detallar la planificación del mismo. Para realizar la planificación se ha utilizado una herramienta básica en la rama de ingeniería del software; el diagrama de Gantt.

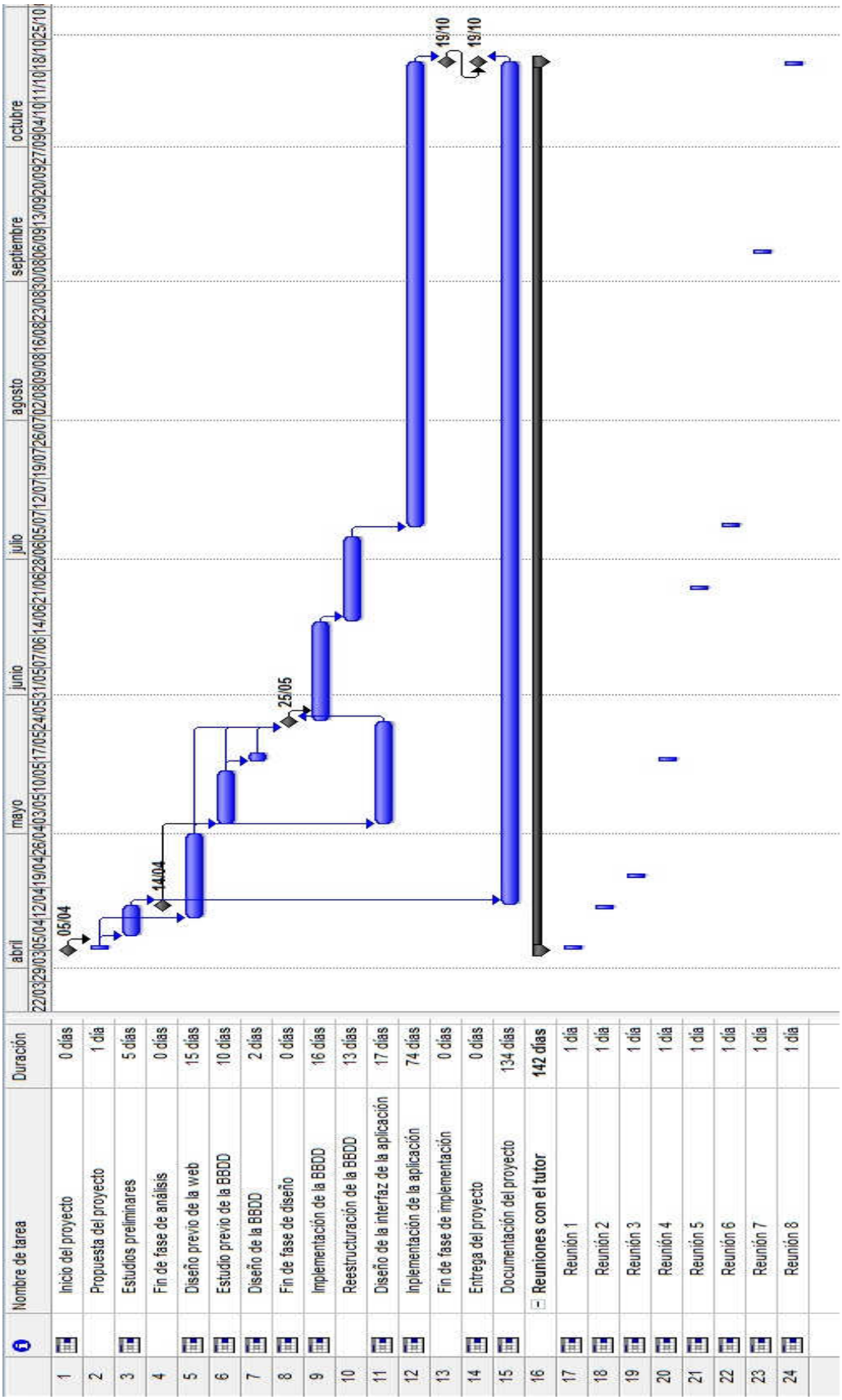
El diagrama de Gantt es una popular herramienta gráfica utilizada para mostrar el tiempo previsto de esfuerzo para realizar un trabajo. A pesar de que, en principio, el diagrama de Gantt no indica las relaciones existentes entre actividades, la posición de cada tarea a lo largo del tiempo hace que se puedan identificar dichas relaciones e independencias. Fue Henry Laurence Gantt quien, entre 1910 y 1915, desarrolló y popularizó este tipo de diagrama en Occidente.

En gestión de proyectos, el diagrama de Gantt se ha convertido en una herramienta básica con la finalidad de representar las diferentes unidades mínimas de trabajo y las fases, tareas y actividades programadas como parte de un proyecto.

A groso modo, el diagrama está compuesto por un eje vertical donde se establecen las actividades que constituyen el trabajo que se va a ejecutar, y un eje horizontal que muestra en un calendario, la duración de cada una de ellas.

Las desviaciones en el calendario han sido habituales a lo largo del periodo de desarrollo. Estas demoras son debidas a la necesidad de compaginar el desarrollo del proyecto con los estudios de grado y por tanto, la realización de exámenes y prácticas que ello conlleva.

En las siguiente página se muestra la figura que contiene la planificación del proyecto completo. Teniendo en cuenta que cada día de trabajo corresponden a cuatro horas reales, excepto en la fase de documentación que corresponde a una hora de trabajo.



	Nombre de tarea	Duración
1	Inicio del proyecto	0 días
2	Propuesta del proyecto	1 día
3	Estudios preliminares	5 días
4	Fin de fase de análisis	0 días
5	Diseño previo de la web	15 días
6	Estudio previo de la BBDD	10 días
7	Diseño de la BBDD	2 días
8	Fin de fase de diseño	0 días
9	Implementación de la BBDD	16 días
10	Reestructuración de la BBDD	13 días
11	Diseño de la interfaz de la aplicación	17 días
12	Implementación de la aplicación	74 días
13	Fin de fase de implementación	0 días
14	Entrega del proyecto	0 días
15	Documentación del proyecto	134 días
16	**Reuniones con el tutor**	**142 días**
17	Reunión 1	1 día
18	Reunión 2	1 día
19	Reunión 3	1 día
20	Reunión 4	1 día
21	Reunión 5	1 día
22	Reunión 6	1 día
23	Reunión 7	1 día
24	Reunión 8	1 día

Ilustración 21: Diagrama de Gantt

Según el diagrama de Gantt , mostrado en la figura anterior, podemos decir que el proyecto realizado ha tenido una duración de 144 días, 538 horas, donde se han realizado las fases de diseño, análisis, implementación y documentación, además, de algunas reuniones presenciales con el tutor del proyecto.

Como aspectos a destacar, resaltar las diferentes tareas de diseño que han sido realizadas en paralelo para reducir tiempo y recursos.

La tarea que ha durado más tiempo ha sido la de documentación puesto que se comenzó con el hito de 'inicio de proyecto' y finaliza al terminar el proyecto. Esto es debido a que en esta fase se recopila la información necesaria para el desarrollo de los objetivos y se van almacenando datos de interés durante todo el tiempo que ha durado la realización.

El calendario laboral utilizado para la planificación es un calendario típico donde los periodos laborables se encuentran de lunes a viernes y los periodos de descanso son los sábados y domingos.

4.2 Presupuesto

En este apartado se va a mostrar de manera detallada el presupuesto desglosado del proyecto, especificando los diferentes gastos que han sido necesarios para su realización.

4.2.1 Resumen de horas dedicadas

Basándonos en el diagrama de Gantt expuesto en apartados anteriores es posible calcular el número de horas totales dedicadas al proyecto.

Fase de análisis: 6 días x 4 horas = 24 horas

Fase de diseño: 19 días x 4 horas = 76 horas

Fase de implementación: 76 días x 4 horas = 304 horas

Fase de documentación: 134 días x 1 hora = 134 horas

Por tanto el número de horas totales dedicadas al proyecto será la suma de las horas dedicadas a cada una de las fases. El coste en horas de la totalidad del proyecto es de 538 horas.

4.2.2 Resumen de personal

En la siguiente tabla se muestran los cargos correspondientes al personal informático cualificado para realizar las distintas tareas o actividades. Los salarios por hora están en consonancia con los salarios de empleados en empresas similares del sector. Todos los costes son calculados sin I.V.A.

Cargo	Nº de horas	Coste Hora	Dedicación (hombre /mes)*	Total (€)
Diseñador	76	25 €	0,95	1900
Ingeniero Senior	24	34,59 €	0,3	830.16
Ingeniero	304	21,72 €	3,8	6602,88
Responsable de documentación	134	17 €	1,675	2278
			TOTAL	**11611,04**

Tabla 48: Resumen del personal

*Utilizando 1 Hombre/ mes = 80 horas según la planificación realizada

4.2.3 Resumen de Hardware

En la siguiente tabla se muestran los equipos informáticos adquiridos con su coste de amortización durante el periodo que dura el proyecto. Todos los costes son calculados sin I.V.A

Descripción	Unidades	Coste (€)	Coste total (€)	Coste imputable
Ordenador Intel® C2D i7 975 2x2Gb 800 CL4	1	800	800	93,33
Teclados Logitech G15	1	30	30	3,50
Ratón Logitech Mx518	1	25	25	2,92
Monitor TFT LG L1960TQ	1	120	120	14,00
Impresora HP Deskjet 6980	1	100	100	1,67
Pendrive Kingston 16 Gb	1	40	40	4,67
			TOTAL	**120,08**

Tabla 49: Resumen de Hardware

Siendo la amortización:

$\frac{A}{B} * C * D$

A = nº de meses desde la fecha de facturación

B = periodo de depreciación (60 meses)

C = coste del equipo

D = % de uso que se dedica al proyecto

4.2.4 Resumen de Software y licencias

En la siguiente tabla se muestran las herramientas software necesarias para el proyecto. Todos los costes son calculados sin I.V.A.

Descripción	Unidades	Coste (€)	Coste total (€)
Micorsoft Office Professional 2007	1	250	250
Microsoft Office Project 2007	1	200	200
Microsoft Office Visio 2007	1	200	200
Editor Notepad++	1	0	0
XAMPP Lite	1	0	0
Mozilla Firefox 3.6.10	1	0	0
Google Chrome	1	0	0
Herramientas de Google Maps	1	0	0
		TOTAL	**650**

Tabla 50: Resumen de software y licencias

4.2.5 Resumen de material fungible

En la siguiente tabla se muestra el material fungible que se estima necesario para la realización del proyecto, así como sus costes. En material de escritorio englobamos folios, bolígrafos, gomas, lápices, clips, carpetas, archivadores, recambios, grapadoras y demás material de oficina. Todos los costes son calculados sin I.V.A.

Descripción	Coste total (€)
Material de escritorio variado	150
Recambios de impresora	100
TOTAL	**250**

Tabla 51: Resumen de material fungible

4.2.6 Resumen del presupuesto total

En la siguiente tabla se muestra el sumatorio de los totales anteriormente calculados pero sin el I.V.A. incluido. A la suma de los costes le vamos a añadir un diez por ciento en concepto de costes indirectos, lo cual equilibrará los riesgos del proyecto y aquellos otros valores que no se han tenido en cuenta al realizar el presupuesto.

Descripción	Coste total (€)
Equipo de trabajo	11.611,04
Amortización	**120**
Subcontratación de tareas	0
Costes de funcionamiento	900
Costes indirectos (20%)	**2.526**
TOTAL	**15.157,04**

Tabla 52: Resumen del presupuesto sin I.V.A.

4.2.7 Plantilla resumen

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Gloria Rivero Ortega
2.- Departamento:
Informática
3.- Descripción del Proyecto:
- Titulo: **Aplicación web para la gestión de redes inalámbricas de sensores**
- Duración (meses): **6,725**
Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):
Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Gloria Rivero Ortega		Diseñador	0,95	2.000,00	1.900,00	
Gloria Rivero Ortega		Ingeniero Senior	0,3	2.767,20	830,16	
Gloria Rivero Ortega		Ingeniero	3,8	1.737,60	6.602,88	
Gloria Rivero Ortega		Documentación	1,675	1.360,00	2.278,00	
					0,00	
		Hombres mes	**6,725**	**Total**	11.611,04	

** Utilizado 1 Hombre mes = 80 horas según la planificación realizada
[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
PC Intel® C2D i7 975 2x2Gb 800	800,00	100	7	60	93,33
Teclados Logitech G15	30,00	100	7	60	3,50
Ratón Logitech Mx518	25,00	100	7	60	2,92
Monitor TFT LG L1960TQ	120,00	100	7	60	14,00
Impresora HP Deskjet 6980	100,00	100	1	60	1,67
Pendrive Kingston 16 Gb	40,00	100	7	60	4,67
				Total	120,08

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} \text{xCxD}$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS - NO APLICA

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Microsoft Office Professional 2007		250,00
Microsoft Visio 2007		200,00
Microsoft Project 2007		200,00
Material fungible		250,00
	Total	900,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	11.611
Amortización	120
Subcontratación de tareas	0
Costes de funcionamiento	900
Costes Indirectos	2.526
Total	15.157

Ilustración 22: Plantilla de presupuesto

Capítulo 5

Conclusiones y Trabajos futuros

En este apartado se exponen las conclusiones que se han obtenido del trabajo realizado, valorando los objetivos cumplidos que se plantearon al principio del proyecto. También se analizan los posibles trabajos futuros que se podrían llevar a cabo para mejorar este proyecto. En último lugar se mostrará la bibliografía utilizada para poder realizar este trabajo.

5.1 Conclusiones

El proyecto comenzó con la idea de mostrar una red de sensores en una aplicación, primeramente se pensó situar dichos sensores en edificios diseñados con herramientas 3D, para especificar en detalle la posición de cada uno de los nodos de la red dentro de los edificios, pero posteriormente se decidió situarlos en un mapa de Google para comprobar las posibilidades y limitaciones de esta herramienta.

Se ha realizado un trabajo intenso en cuanto al estudio de documentación acerca de las API de Google y la manera de conseguir alcanzar los objetivos propuestos mediante esta herramienta gratuita. En este trabajo nos dimos cuenta del gran número de posibilidades que existen para tratar los mapas e incluirlos en la aplicación, pero también de algunas limitaciones inherentes a la propia herramienta que no permitían hacer lo deseado en el proyecto.

La principal limitación fue la del uso de las pestañas de los mapas, pues solo se permite código HTML y por tanto, la idea principal, que consistía en situar los gráficos de Google Chart en esas pestañas, tuvo que ser descartada. De este modo, se inició un nuevo estudio sobre tecnologías de mercado mediante llamadas en HTML hasta que se encontró JPGraph, que hizo posible seguir con nuestra labor de dibujar gráficos en las pestañas de los mapas de Google Maps.

Una vez resuelto este inconveniente nos embarcamos en la tarea de poder actualizarlos mostrando datos en tiempo real. En este caso, el problema era que cuando se realizaba una consulta a la base de datos para recopilar nuevos valores y mostrarlos en la aplicación, la página se tenía que refrescar en su totalidad, lo que disminuía notablemente la usabilidad, así que se decidió aprender a utilizar la tecnología Ajax para solucionar esto.

Podríamos decir que estas han sido las principales dificultades a resolver durante el desarrollo del proyecto, sin mencionar la primera de todas, en la fase de análisis cuando se vio la necesidad de recurrir a un software que realizara las funciones de servidor para poder unir la base de datos con la aplicación.

Todos estos inconvenientes encontrados por el camino han sido resueltos, creo, que de manera positiva y satisfactoria, pues se han cumplido los objetivos propuestos al inicio del proyecto además del valor añadido que supone haber aprendido a usar tecnologías que me resultaban totalmente desconocidas y que actualmente tienen mucha relevancia como son Ajax, PHP y el paquete XAMPP.

5.2 Trabajos futuros

Teniendo en cuenta que el objetivo principal del proyecto es investigar sobre posibles representaciones gráficas de redes de sensores inalámbricas, nuestra aplicación es un prototipo de posibles soluciones que se pueden llevar a cabo, por tanto, a partir de este proyecto se pueden realizar mejoras para aumentar las funcionalidades.

- Base de datos:

  - Almacenar la base de datos en un servidor web, para centralizar los datos y conseguir acceso a ellos desde distintos dispositivos.
  - Implementar la base de datos a partir de otros sistemas gestores para poder realizar comparativas de accesibilidad, fiabilidad y eficacia.

- Aplicación web:

  - Mejorar la accesibilidad del portal web para aquellos usuarios con algún tipo de discapacidad.

  - Realizar un control de acceso, permitiendo a los administradores tener control sobre quien puede ver y administrar el contenido.

  - Aumentar la aplicación de manera de que a partir de cada gráfico se obtengan estadísticas y valores aclaratorios para mantener el buen funcionamiento de cada nodo de la red.

- Otras mejoras:

  - Desarrollar la aplicación para dispositivos móviles para que de este modo, se aumente la efectividad de la aplicación, pues en caso de emergencia, el aviso lo recibirá de inmediato el encargado de las redes de sensores.

5.3 Bibliografía extra utilizada

Barriocanal, Luis. (2008). Introducción a los sitios web dinámicos*. Observatorio Tecnológico-Ministerio de Educación, Política Social y Deporte- Gobierno de España.*

Chapman, Cameron.(2009). The Evolution of Web Design. Six Revisions- Web Development and Design Information.
http://sixrevisions.com/web_design/the -evolution-of-web-design/.

Departamento de bases de datos.**UC3M**. Material didáctico de Diseño de Bases de Datos. conocimientos y técnicas avanzadas para el diseño de bases de datos y acceso a las mismas.
http://labda.inf.uc3m.es/doku.php

Google. APIS, Code, Todo acerca de herramientas de desarrollo.
http://code.google.com/intl/es/apis/ajax/
http://code.google.com/intl/es/apis/maps/
http://code.google.com/intl/es/apis/charttools/index.html

Lenguajes de programación y herramientas. Información general.
http://es.wikipedia.org

Piattini, José A.Calvo-Manzano, Joaquín Cervera, Luis Fernandez. (2003). Aplicaciones informáticas de gestión. Ingeniería del Software. Modelo en espiral.

The design of site-Paterns. Material didáctico de Sistemas Hipermedia: técnicas de diseño web, manejo de lenguajes de programación, aplicaciones de patrones de diseño y evaluación de sitios web.

Tutoriales de Ajax. Programación
www.librosweb.es/ajax/
www.maestrosdelweb.com/editorial/ajax
www.w3schools.com/ajax/default.asp

Tutoriales de JPGraph. Programación
jpgraph.net/
blog.unijimpe.net/jpgraph-graficos-con-php/
http://www.desarrolloweb.com/articulos/1987.php

Tutoriales de PHP. Programación
http://www.programacion.com/php/tutorial/php/
http://www.php.net/manual/es/book.mysql.php

Tutoriales de SQL. Programación
www.mysql.com/
dev.mysql.com/doc/refman/5.0/es/index.html

WorldWideWebConsortium (W3C)- Oficina Española. 2008.*World Wide Web Consortium* (W3C) *http://www.w3c.es.*

XAMPP Portable. The complete, portable server
http://portableapps.com/apps/development/xampp

XAMPP Apache Friends. Sitio web acerca de la instalación y uso del paquete.
http://www.apachefriends.org/es/xampp.html

Capítulo 6

Anexos

En este último capítulo se adjuntan los anexos explicativos del desarrollo y herramientas que han sido citadas durante todo el documento. En ellos se adjunta código utilizado a modo aclarativo.

Anexo 1: Manual de instalación y uso del servidor

1.1 Sobre este manual

Este es el manual de referencia para el servidor de base de datos MySQL, versión 5.0. No está destinado para usarse con versiones más antiguas del software MySQL debido a las numerosas diferencias funcionales y de otro tipo entre MySQL 5.0 y versiones previas.

Este manual es una referencia, por lo que no proporciona instrucciones sobre conceptos generales de SQL o de bases de datos relacionales. Tampoco enseña sobre cómo usar su sistema operativo o su intérprete de línea de comandos.

1.2 Información general

El software MySQL® proporciona un servidor de base de datos SQL muy rápido. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido. MySQL es una marca registrada de MySQL AB.

El software MySQL tiene una doble licencia. Los usuarios pueden elegir entre usar el software MySQL como un producto Open Source bajo los términos de la licencia GNU o pueden adquirir una licencia comercial estándar de MySQL AB.

1.1 Intalación

El primer paso es descargar el paquete de aplicaciones XAMPP en su versión portable, cuyo enlace de descarga es el siguiente:

http://portableapps.com/apps/development/xampp

Si se prefiere instalar la versión convencional de la aplicación, hay que recurrir al portal oficial de XAMPP (*http://www.apachefriends.org/es/xampp.html*). Para instalar esta versión no será necesario seguir los pasos de este apartado.

Cuando el archivo se haya descargado, se deberá ejecutar y se instalará de manera automática con el asistente de instalación convencional.

El primer paso es arrancar el servidor, para ello hay que dirigirse a la carpeta creada tras la instalación y ejecutar el archivo "*setup_xampp.bat*". Este archivo cambia la unidad asignada al servidor para que coincida con la que tiene asignado el dispositivo portátil en el que se haya instalado. Posteriormente se arranca el servidor ejecutando el archivo "xampp_start.exe".

Si todo este proceso ha ido correctamente, el servidor ya está funcionando. Ahora diríjase al navegador Web y teclee la siguiente dirección:

http://localhost/phpmyadmin

Desde esta dirección se accede a la herramienta que actúa de interfaz con el sistema gestor de bases de datos y gracias a la cual se podrá crear de manera sencilla la base de datos necesaria.

En la página de inicio de la herramienta aparece un cuadro que nos permite crear una base de datos. El cuadro se ha de rellenar como se muestra en la siguiente figura:

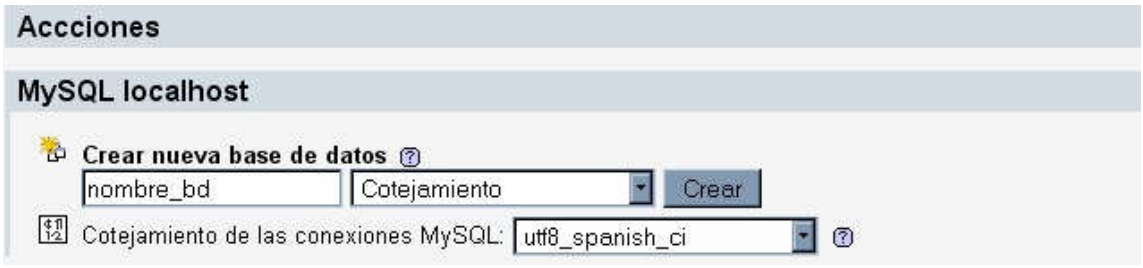


Ilustración 23: Crear base de datos en phpMyAdmin

El siguiente paso es asignarle una contraseña al usuario "root" de esta base de datos. Para ello hay que seleccionar la base de datos que se acaba de crear y dirigirse a la sección *privilegios*. En esta sección se editarán los datos del usuario y se le añadirá una contraseña.

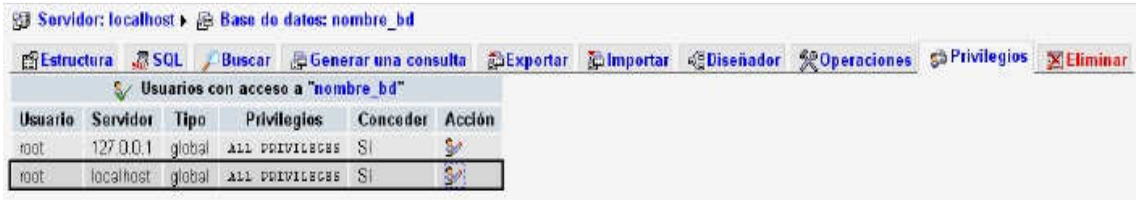


Ilustración 24: Configuración de usuario en phpMyAdmin

Como último paso hay que modificar el fichero de configuración de phpMyAdmin para añadir el cambio de contraseña, si no, phpMyAdmin no funcionará. Para ello hay que acceder al archivo situado en la ruta (*/xampp/phpMyAdmin/config.inc.php*). En la siguiente ilustración se muestra el contenido del archivo, en el cual, habrá que escribir el en cuadrado negro la contraseña asignada.

```
18  /* Authentication type and info */
19  $cfg['Servers'][$i]['auth_type'] = 'config';
20  $cfg['Servers'][$i]['user'] = 'root';
21  $cfg['Servers'][$i]['password'] = 
22  $cfg['Servers'][$i]['AllowNoPasswordRoot'] = true;
```

Ilustración 25: Fichero config.inc.php

En este momento la instalación ha finalizado. A partir de este momento se podrá crear la base de datos y ejecutar desde phpMyAdmin.

Anexo 2: Acerca de la base de datos

2.1 Creación de tablas

Este anexo está dedicado al script SQL de creación de bases de datos. Después de haber explicado las peculiaridades de la implementación, se anexa el script completo para resolver dudas o curiosidades en caso que fuera necesario.

```
CREATE TABLE `pfc`.`Marcador` (
`Latitud` DECIMAL NOT NULL ,
`Longitud` DECIMAL NOT NULL ,
`Id_Zona` VARCHAR( 3 ) NOT NULL ,
`Color` VARCHAR( 20 ) NULL ,
PRIMARY KEY ( `Latitud` , `Longitud` ),
FOREIGN KEY (`Id_Zona`) REFERENCES zona (`Id_Zona`)
ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB;
```

```
CREATE TABLE `pfc`.`Datos` (
`Timestamp` TIMESTAMP NOT NULL ,
`Id_Sensor` VARCHAR( 3 ) NOT NULL ,
`Medicion` FLOAT NOT NULL ,
PRIMARY KEY ( `Timestamp` , `Id_Sensor` ),
FOREIGN KEY (`Id_Sensor`) REFERENCES sensor (`Id_Sensor`)
ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB;
```

```
CREATE TABLE `pfc`.`Sensor` (
`Id_Sensor` VARCHAR( 3 ) NOT NULL ,
`Tipo` VARCHAR( 20 ) NOT NULL ,
`Umbral_min` SMALLINT NOT NULL ,
`Umbral_max` SMALLINT NOT NULL ,
`Alarm_min` SMALLINT NULL ,
`Alarm_max` SMALLINT NULL ,
`Horas_Autonomia` SMALLINT NOT NULL ,
`Fecha_Revision` DATE NOT NULL ,
`Latitud` DECIMAL NOT NULL ,
`Longitud` DECIMAL NOT NULL ,
PRIMARY KEY ( `Id_Sensor` ),
FOREIGN KEY (`Latitud`,`Longitud`) REFERENCES marcador (`Latitud`,`Longitud`)
ON DELETE CASCADE ON UPDATE CASCADE) ENGINE = InnoDB;
```

```
CREATE TABLE `pfc`.`Responsable` (
`DNI` VARCHAR( 9 ) NOT NULL ,
`Nombre` VARCHAR( 20 ) NOT NULL ,
`Apellidos` VARCHAR( 40 ) NOT NULL ,
`Telefono` INT( 9 ) NOT NULL ,
`Email` VARCHAR( 30 ) NOT NULL ,
PRIMARY KEY ( `DNI` )
) ENGINE = InnoDB ;
```

```
CREATE TABLE `pfc`.`Zona` (
`Id_Zona` VARCHAR( 3 ) NOT NULL ,
`Nombre` VARCHAR( 25 ) NOT NULL ,
`CP` SMALLINT( 5 ) UNSIGNED NOT NULL ,
`Provincia` VARCHAR( 25 ) NOT NULL ,
`Localidad` INT( 25 ) NOT NULL ,
`Tipo` ENUM( 'Urbana', 'Rural' ) NOT NULL ,
`Id_Responsable` VARCHAR( 3 ) NULL ,
`Direccion_Web` VARCHAR( 30 ) NOT NULL ,
`Descripcion` TEXT NULL ,
PRIMARY KEY ( `Id_Zona` ),
FOREIGN KEY (`Id_Responsable`) REFERENCES responsable (`DNI`)
ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE = InnoDB

CREATE TABLE `pfc`.`Urbana` (
`Id_Zona` VARCHAR( 3 ) NOT NULL ,
`Direccion` VARCHAR( 40 ) NOT NULL ,
PRIMARY KEY ( `Id_Zona` ) ,
FOREIGN KEY (`Id_Zona`) REFERENCES zona (`Id_Zona`)
ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB;

CREATE TABLE `pfc`.`Rural` (
`Id_Zona` VARCHAR( 3 ) NOT NULL ,
`Superficie_m2` MEDIUMINT UNSIGNED NOT NULL ,
`Tipo` ENUM( 'Cultivo', 'Parque Forestal', 'Reserva Natural' ) NOT NULL ,
PRIMARY KEY ( `Id_Zona` ) ,
FOREIGN KEY (`Id_Zona`) REFERENCES zona (`Id_Zona`)
ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB;
```

2.2 Diccionario de datos

El diccionario de datos es la lista de todos los elementos que forman parte del flujo de datos de todo el sistema, en este se guardan los detalles y descripción de todos estos elementos.

Tabla Datos:

Campo	Tipo	Nulo	Predeterminado	Enlaces a
Timestamp	timestamp	No	CURRENT_TIMESTAMP	
Id_Sensor	varchar(3)	No		Sensor -> Id_Sensor
Medicion	float	No		

Tabla 53: Diccionario de datos de la tabla Datos

Tabla Rural:

Campo	Tipo	Nulo	Predeterminado	Enlaces a
Id_Zona	varchar(3)	No		Zona -> Id_Zona
Superficie_m2	medium(8)	No		
Tipo	Enum('Cultivo', 'Parque forestal', Reserva natural'))	No		

Tabla 54: Diccionario de datos de la tabla Rural

Tabla Urbana:

Campo	Tipo	Nulo	Predeterminado	Enlaces a
Id_Zona	varchar(3)	No		Zona -> Id_Zona
Direccion	varchar(40)	No		

Tabla 55: Diccionario de datos de la tabla Urbana

Tabla Sensor:

Campo	Tipo	Nulo	Predeterminado	Enlaces a
Id_Sensor	varchar(3)	No		
Tipo	varchar(20)	No		
Umbral_Min	smallint(6)	No		
Umbral_Max	smallint(6)	No		
Alarm_Min	smallint(6)	Si	Null	
Alarm_Max	smallint(6)	Si	Null	
Horas_Autonomia	smallint(6)	No		
Fecha_Revision	date	No		
Latitud	decimal (10,6)	No		Marcador -> Latitud
Longitud	decimal (10,6)	No		

Tabla 56: Diccionario de datos de la tabla Sensor

Tabla Marcador:

Campo	Tipo	Nulo	Predeterminado	Enlaces a
Latitud	decimal (10,6)	No		
Longitud	decimal (10,6)	No		
Id_Zona	varchar(3)	No		Zona -> Id_Zona
Color	varchar(20)	Si	Null	

Tabla 57: Diccionario de datos de la tabla Marcador

Tabla Responsable:

Campo	Tipo	Nulo	Predeterminado	Enlaces a
DNI	varchar(9)	No		
Nombre	varchar(20)	No		
Apellidos	varchar(40)	No		
Telefono	int(9)	No		
Email	varchar(30)	No		

Tabla 58: Diccionario de datos de la tabla Responsable

Tabla Zona:

Campo	Tipo	Nulo	Predeterminado	Enlaces a
Id_Zona	varchar(3)	No		
Nombre	varchar(50)	No		
CP	smallint(5)	No		
Provincia	varchar(25)	No		
Localidad	varchar(25)	No		
Tipo	Enum ('Urbana', 'Rural')	No		
Id_Responsable	varchar(9)	Si	Null	Responsable -> DNI
Direccion_Web	varchar(30)	No		
Descripcion	text	Si	Null	

Tabla 59: Diccionario de datos de la tabla Zona

Anexo 3: Acerca de la aplicación web

La aplicación no ha sido diseñada para su uso por cualquier tipo de usuario, es un tipo de aplicación destinada a organizaciones de control y mantenimiento de redes de sensores inalámbricas. Por tanto, se ha intentado hacer un diseño adecuado al tipo de usuario, facilitando la accesibilidad.

La portada es intuitiva para que el usuario se posicione según el tipo de zona que desea comprobar:

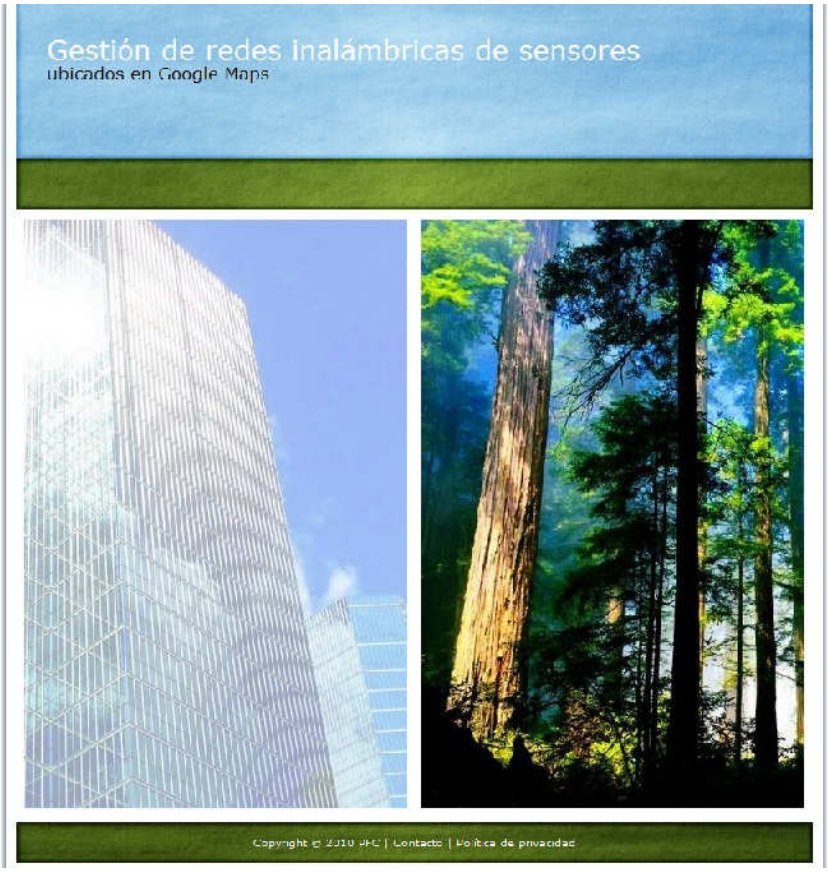


Ilustración 26: Portada de la aplicación

La imagen toma color, cuando se pasa el ratón por encima, de este modo se selecciona el tipo de zona.

Una vez seleccionada el tipo de zona se accede a una pantalla intermedia para elegir la zona del tipo escogido que se desea visualizar:

Ilustración 27: Menú de Zonas urbanas

Aún estando en el menú de un tipo de zona, se puede acceder al otro tipo en cualquier momento, gracias al menú superior, de este modo se facilita la corrección de errores por parte del usuario final.

Una vez que nos encontremos en la parte concreta de una zona se podrá visualizar la descripción de la misma así como el mapa situado en el punto a gestionar. La descripción de la zona se obtiene de la base de datos, para facilitar la modificación de los mismos en caso necesario. Del mismo modo, los marcadores posicionados en el mapa obtienen sus datos gracias al dinamismo creado en la aplicación y por tanto será necesario modificar la base de datos para realizar cambios en la misma.

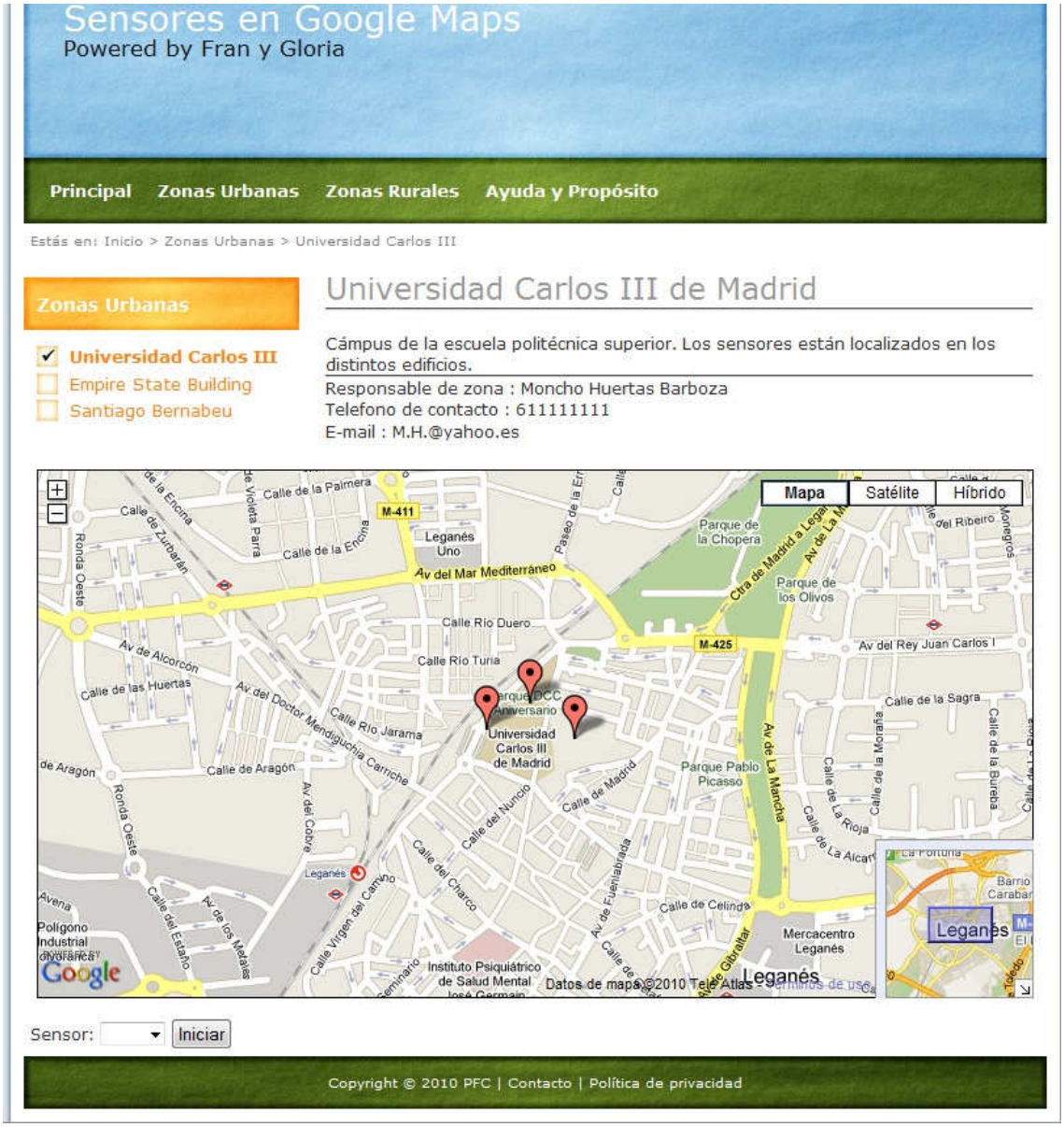


Ilustración 28: Página principal de una zona

Desde cada una de las zonas se puede acceder fácilmente a las demás implantadas en la aplicación, ya sean del mismo tipo, mediante el menú lateral izquierdo, o del otro tipo de zonas posibles, mediante el menú superior. Además de las migas de pan para evitar la pérdida del usuario por la navegación y facilitar el acceso a los pasos previos realizados. El resto de las zonas tienen el mismo formato que el de la figura anterior.

Utilizando el mapa interactivo podemos tener acceso a cada marcador haciendo clic en él, de esta manera, la aplicación nos muestra a través de un globo informativo dividido en pestañas, los datos generado por los sensores que se encuentran en el marcador seleccionado.

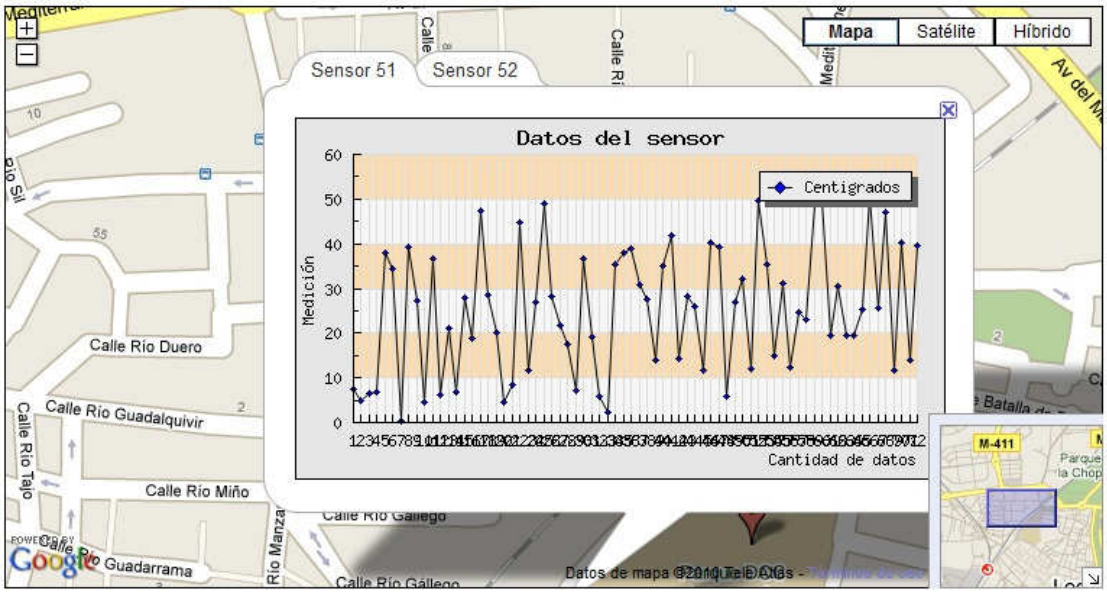


Ilustración 29: Información del marcador

Además de las páginas para la gestión de las redes inalámbricas de sensores, se encuentran enlaces en el pié de página para acceder a zonas informativas sobre el mantenimiento de la aplicación. Estas páginas son estáticas para cumplir una finalidad meramente informativa.

Ilustración 30: Página de contacto

Ilustración 31: Página de privacidad

Ilustración 32: Página de ayuda y propósito